Published in IET Communications
Received on 2nd October 2011
Revised on 21st February 2012
doi: 10.1049/iet-com.2011.0727

ISSN 1751-8628

Adaptive power management system for mobile multimedia device

S.O. Park[1] *J.K. Lee*[1] *J.H. Park*[2] *S.J. Kim*[1]

[1]School of Computer Science and Engineering, Chung-Ang University, 221 Huksuk-Dong, DongJak-Ku, Seoul 156-756, South Korea
[2]Department of Computer Science and Engineering, Seoul National University of Science and Technology, 172 Gongreung 2-dong, Nowon-gu, Seoul 139-742, South Korea
E-mail: sj1st@cs.cau.ac.kr

Abstract: Recently, methods that combine the concepts of dynamic voltage scaling (DVS) and dynamic power management (DPM) approaches have been proposed for a single task. This study proposes adaptive power management system (APMS), which utilises useful features of both DPM and DVS and an existing pattern analysis algorithm, and new break even time-based task partition scheduling (BET-BTPS). APMS splits tasks based on BET to reduce the total power consumption of multiple multimedia applications in a mobile embedded system environment and analyses their usage patterns of peripheral devices and apply the results to their scheduling for further reduction of power consumption. It also takes into account the additional time delay that may occur while invoking and executing a timeslot scheduler to guarantee real-time services for multimedia data streams when the scheduling of APMS is in effect. In order to determine the optimal processor speed that minimises power consumption, the extra consumption required by the scheduler is compared with that reduced by BTPS. The proposed approach demonstrates better performance compared to existing power management schemes not only for single task but multiple tasks.

1 Introduction

Power consumption of an embedded system is determined by the hardware that constitutes the system and the software that drives it. However, even though the hardware supports low power consumption, power-saving effects cannot be maximised if the hardware is not utilised at the level of software such as operating systems. Conventionally, researches to deal with power consumption are focused on the simulation of energy management [1] or direct control of power state [2]. Dynamic voltage scaling (DVS) [3–6] and dynamic power management (DPM) [7, 8] have been widely used for power management. However, DVS attempts to vary the supply voltages of processor cores to control their power consumption, while DPM adjusts the amount of power supplied to a particular device based on the usage patterns of all components except processor cores in a computer system, as long as execution constraints required by its different power states are met. Since these methods are applicable to only either processor cores or peripheral devices but not both, they are limited in reducing power consumption. Recently, researchers have investigated new methods that take into account both processor cores and peripheral devices [9]. However, they have attempted to reduce the power consumption of a single multimedia application and failed to make use of usage patterns of devices used by multimedia applications. In this paper, we assume that a multimedia data stream is partitioned into a sequence of substreams so that each stream can be processed per second to meet a given bit rate of multimedia data. A substream is in turn partitioned into several pieces called timeslots. The length of a timeslot will be the maximum of the break even times (BETs) of peripheral devices.

This paper proposes adaptive power management system (APMS), which utilises useful features of both DPM and DVS and an existing pattern analysis algorithm [10]. APMS also utilises new BET-based task partition scheduling (BTPS). Furthermore, APMS takes into account the additional time delay that may occur while invoking and executing a timeslot scheduler to guarantee real-time services for multimedia data streams when the scheduling of APMS is in effect. In order to determine the optimal processor speed that minimises power consumption, the extra consumption required by the scheduler should be compared with that reduced by BTPS.

The paper proceeds as follows: After discussing related works in Section 2, we explain how to design APMS in Section 3. Section 4 compares the performance of APMS with that of the existing power management schemes, and Section 5 concludes our work and suggests some future research directions.

2 Related works

Conventionally, DPM changes the voltage states of a peripheral device if it has been in the idle state for longer than a predetermined time. Therefore in order to reduce the power consumed by the peripheral device during idle time in DPM, it is very important to predict its idle cycles that will occur later. Conventional DPM adopts the device-level power management (DLPM) mechanism that requires power management at the level of peripheral devices. Recently, task-based power management (TBPM) [11] utilises operating system directed power reduction and analyses peripheral device usage pattern at the task level to optimise power consumption for a single task. Each task informs the task scheduler with its own characteristics in device utilisation. The scheduler then rearranges the execution order of tasks so that changes in power states can be avoided to reduce overhead as far as possible. When a processor core is dynamically able to control its own voltage, DVS is more efficient than DPM in terms of processor power consumption. Thus, DPM is mainly used in peripheral devices.

Low-power fixed priority scheduling (LPFPS) [12] is a hybrid method that uses both DPM and DVS. It is based on a pre-emptive fixed priority scheduling method called rate monotonic (RM) [13]. LPFPS changes voltages dynamically when the supply voltages of processors can be controlled, and operates processors in the power-down mode to reduce energy consumption when the idle periods of processors are considered to be sufficiently long. Although it demonstrates better performances than DPM and DVS, it cannot fully utilise workload-variation slack times (WVSTs), which occur between the execution time and the worst-case execution time (WCET) of a task, because the supply voltage cannot be dynamically adjusted inside a task.

To resolve the LPFPS's problem, run-time voltage hopping (RVH) [14], which uses the timeslot concept, has been proposed. In order to compensate for the overhead due to power state change of peripheral devices, the period during which peripheral devices operate in the power-down mode should be sufficiently long. This period is called the BET of peripheral devices. Even though RVH efficiently utilises the WVST through timeslots, it does not take into account time delays resulted from frequent wakeups of peripheral devices and additional power consumption at the time of wakeups.

Most recently, a research [15] is investigated to deal with power consumption for multimedia application. However, this approach adopted only DVS. In this paper, we propose a power management system that considers multiple tasks. In addition, we attempt to resolve the problems of previous researches, including time delays resulted from frequent wakeups and additional power consumption.

3 APMS

APMS is a power management system that consists of analysis module, scheduling module, control module, execution module and repository, as shown in Fig. 1.

3.1 Analysis module

In order to reduce power consumption of peripheral devices by adjusting the supply voltage, this module utilises the list of peripheral devices used by a delivered timeslot from the execution module to analyse their run-time usage patterns using expectation–maximisation (EM) algorithm [10]. EM algorithm utilises iterative method for finding maximum likelihood or maximum a posteriori among the input patterns, when the algorithm measures the similarity among patterns. Depending on the types of multimedia files

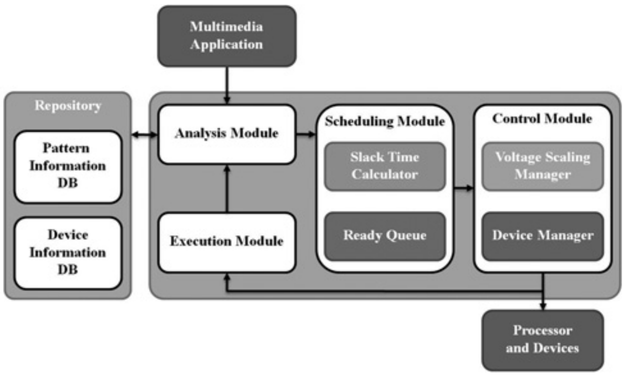

Fig. 1 *Structure of APMS*

utilised by multimedia applications such as MPlayer [16] and their bit rates, the representative usage patterns are created and saved for later use through the pattern analysis algorithm. To be specific, when an application has been executed, the module retrieves patterns from pattern information DB in the repository before requests a multimedia file and checks if there is a representative pattern with the same file type and bit rate. If that is the case, the existing pattern will be retrieved to avoid additional overheads involved in pattern analysis.

A significant amount of disk space may be required to collect and save information on peripheral devices in use. To reduce such storage requirements, rather than searching representative patterns after collecting all pattern information, current representative patterns are revised every time new pattern information is collected until they are determined. To revise undetermined representative patterns, we utilise the usage counts of individual peripheral devices and the total usage counts of the entire peripheral devices in EM algorithm. If revised undetermined representative patterns satisfy the reference value of EM algorithm, they will be finalised as representative patterns. Once they are chosen, no more patterns will be collected in order to reduce storage requirements.

3.2 Scheduling module

This module consists of the slack time calculator (STC) and the ready queue (RQ). BTPS adopted by this module partitions a substream of the multimedia data stream into a sequence of timeslots, so that the WVST can be fully utilised. To be specific, if an idle time of a device is longer than its BET, its state can be changed to 'idle' to reduce its power consumption, which can be lowered so far as the duration of a device's idle time is longer than the maximum of the BETs of its peripheral devices. This module also checks if a device will be used by more than one timeslot consecutively. If so, it schedules those timeslots so that unnecessary sleeping can be prevented since power consumption is greatly affected by its number of wakeups. The processor speed may be lowered if it does not violate the deadline of a timeslot by analysing its slack time using the STC. Equation (1) calculates the total slack time (τ_s), where τ_d and τ_e are the deadline and execution time, respectively, for a given timeslot.

$$\tau_s = \tau_d - \tau_e \tag{1}$$

The RQ manages timeslots generated from multiple multimedia data streams to be processed. When it determines the next timeslot to be processed, it identifies the timeslots during which a specific device is utilised either simultaneously or consecutively. It then modifies the processing sequence of timeslots to minimise changes in the device's power state (on/off) and to maximise the total slack time of timeslots so that power consumption can be reduced as much as possible as shown in Fig. 2.

When determining a next task to be executed from the RQ, the module chooses another task, which utilises the same device(s) as Timeslot 2 of Task 1 and Timeslot 1 of Task 2 in Fig. 3*a* due to the processor idle times. If Timeslot 2 of Task 1 and Timeslot 1 of Task 2 that use the same devices are not scheduled consecutively, as shown in Fig. 3*b*, then devices 5 and devices 6 must remain continuously in the idle state or their power should be turned off and then on again, resulting in more power consumption. Therefore we need to check if there are timeslots that use one or more identical devices consecutively. If so, the timeslots should be scheduled as shown in Fig. 3*a* so that idle times of devices can be maximised by minimising their state changes; otherwise, they should be scheduled as shown in Fig. 3*c*.

3.3 Control module

This module consists of voltage scaling manager (VSM) and device manager (DM). VSM attempts to minimise power consumption by changing the processor speed through adjustment of supply voltage of a processor core using the DVS scheme. APMS determines the processor speed using (2)–(5). Table 1 shows a brief description of each notation used in the equations.

Assuming that the clock cycles required for processing a timeslot follow the uniform distribution, (2) calculates the minimum processor speed required to guarantee a given deadline of the timeslot.

$$\frac{1}{f_{\text{scal}}}\left(\frac{\tau_{\text{w}}+\tau_{\text{b}}}{2}+\tau_{\text{o}}\right)\leq\tau_{\text{d}} \tag{2}$$

Equation (3) shows the minimum processor speed which is required to meet all the deadlines of timeslots belonging to multiple multimedia data streams, taking account of τ_{o}.

$$f_{\text{scal}}\geq\left(\frac{\tau_{\text{w}}+\tau_{\text{b}}+2\tau_{\text{o}}}{2\tau_{\text{d}}}\right) \tag{3}$$

Equation (4) can be used to check if there is any reduction in power consumption required for executing APMS while

```
If a next timeslot is the last timeslot of a current task (CT)
  If {$d_i$|$i$ = 1,..., $n$, where a device $d_i$ has been used in the current timeslot} ∩ {$d_j$ | $j$ = 1,..., $m$, where a device $d_j$ has been used in the last timeslot} = Ø
    If {$d_j$ | $j$ = 1,..., $m$, where a device $d_j$ has been used in the last timeslot of CT} ∩ {$d_k$ | $k$ = 1,..., $l$, where a device $d_k$ has been used in the current timeslot of next task} ≠ Ø
      Delay the schedule of the last timeslot for a period of (CT's deadline – CT's WCET)
  Else
    Execute the schedule of the last timeslot immediately;
Else
  Execute the schedule of the next timeslot immediately;
```

Fig. 2 *Algorithm*

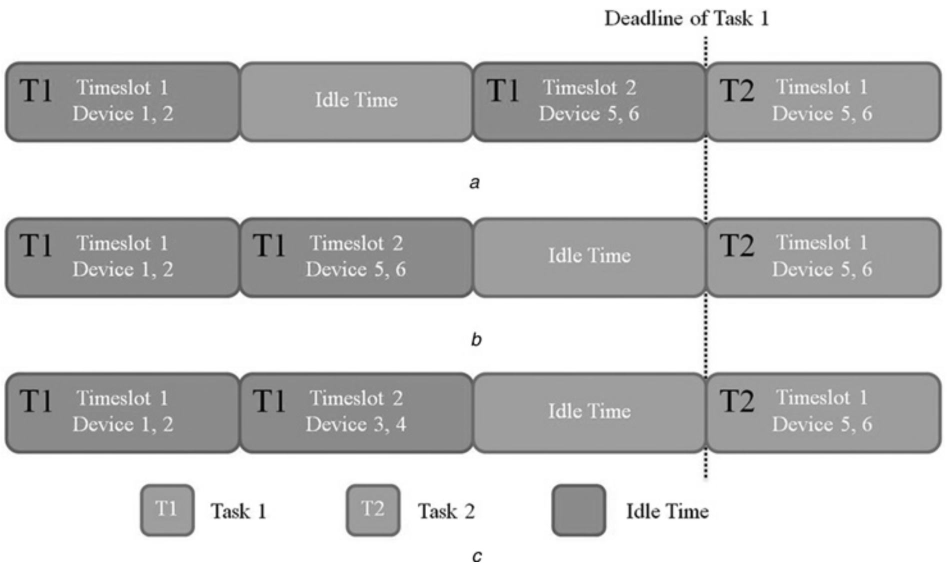
Deadline of Task 1
T1 Timeslot 1 Device 1, 2
Idle Time
T1 Timeslot 2 Device 5, 6
T2 Timeslot 1 Device 5, 6
a
T1 Timeslot 1 Device 1, 2
T1 Timeslot 2 Device 5, 6
Idle Time
T2 Timeslot 1 Device 5, 6
b
T1 Timeslot 1 Device 1, 2
T1 Timeslot 2 Device 3, 4
Idle Time
T2 Timeslot 1 Device 5, 6
T1 Task 1
T2 Task 2
Idle Time
c

Fig. 3 *Different execution orders of timeslots when the same devices are consecutively used*

Table 1 Table of notations

Symbol	Meaning
f_{scal}, f_{max}	the processor speed adjusted by voltage scaling and the maximum processor speed, respectively (Hertz)
τ_d, τ_e, τ_w, τ_b	the deadline, execution, worst execution and best execution time of a timeslot, respectively (clock cycle)
τ_o	the execution time required to invoke the scheduling module due to data stream partitioning (clock cycle)
γ_{p0}, γ_{d0}	power consumption of a processor and peripheral devices during timeslot processing, respectively (Watt)
γ_{p1}, γ_{d1}	power consumption of a processor and peripheral devices during timeslot scheduling, respectively (Watt)
γ_{p2}, γ_{d0}	power consumption of a processor and peripheral devices after timeslot scheduling, respectively (Watt)

processing a single timeslot.

$$f_{\max}\tau_e(\gamma_{p0}+\gamma_{d0}) - f_{\text{scal}}\tau_e(\gamma_{p2}+\gamma_{d2}) \geq f_{\text{scal}}\tau_o(\gamma_{p1}+\gamma_{d1}) \quad (4)$$

The first and second terms of the left-hand side of (4) represent power consumption when no power management method is employed and when our method is employed, respectively. The right-hand side of (4) represents the extra power consumption required by our method. If the left-hand side is greater than or equal to the right-hand side, we may conclude that power consumption of a multimedia application employing our method is less than that not employing any power management method whatsoever.

$$f_{\text{scal}} \leq \frac{f_{\max}\tau_e(\gamma_{p0}+\gamma_{d0})}{\tau_e(\gamma_{p2}+\gamma_{d2})+\tau_o(\gamma_{p1}+\gamma_{d1})} \quad (5)$$

Equation (5) derived from (4) can be used to calculate the maximum value of f_{scal}. APMS guarantees all the deadlines of applications and reduces their power consumption, since the range of the processor speed f_{scal} can be determined by (3) and (5) which consider deadline of timeslot.

3.4 Repository

The repository consists of pattern information DB and device information DB. The former contains representative usage pattern information for multimedia applications such as the types and bit rates of multimedia files. The latter contains inherent attributes of each peripheral device such as its state (e.g. on/off, run, idle, wakeup) and power consumption, which are gathered by and delivered from the analysis module. It also contains the BET of a device that can be obtained from power consumption required during which the device is in one of the three states: run, idle, wakeup. The BET can be calculated by (6). Table 2 explains notations used in (6).

$$\text{BET} = \frac{2\times\tau_w\times(\gamma_w-\gamma_r)}{\gamma_r-\gamma_i} \quad (6)$$

Table 2 Table of notations

Symbol	Meaning
τ_w	the wake-up time of a peripheral device
γ_w	the power consumed to wake up a peripheral device
γ_r	the power consumed by an operating peripheral device
γ_i	the power consumed by an idle peripheral device

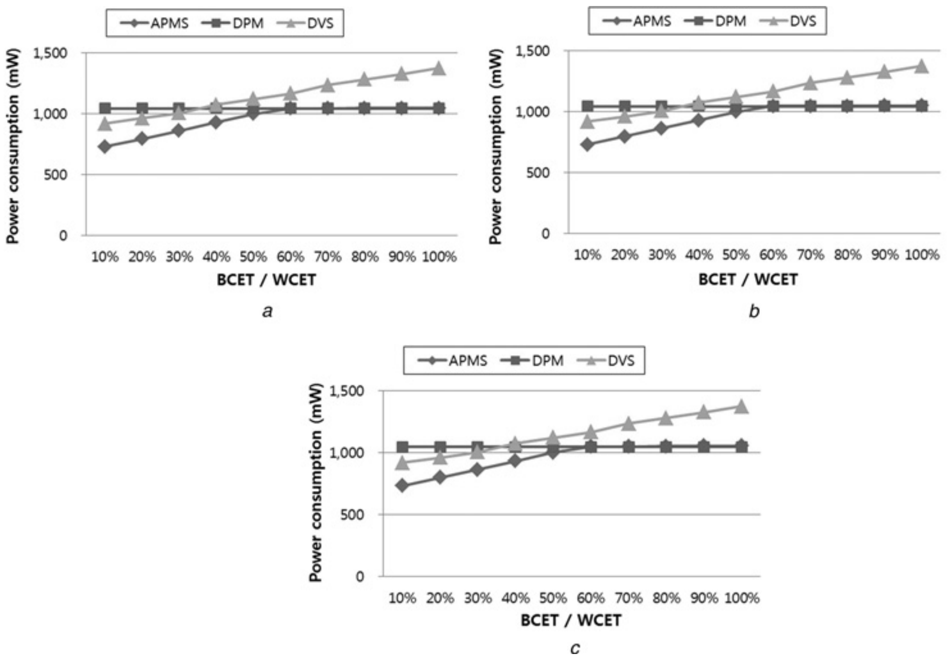

Fig. 4 *Power consumption of a single task for different bit rates*

a 32 kb/s
b 192 kb/s
c 448 kb/s

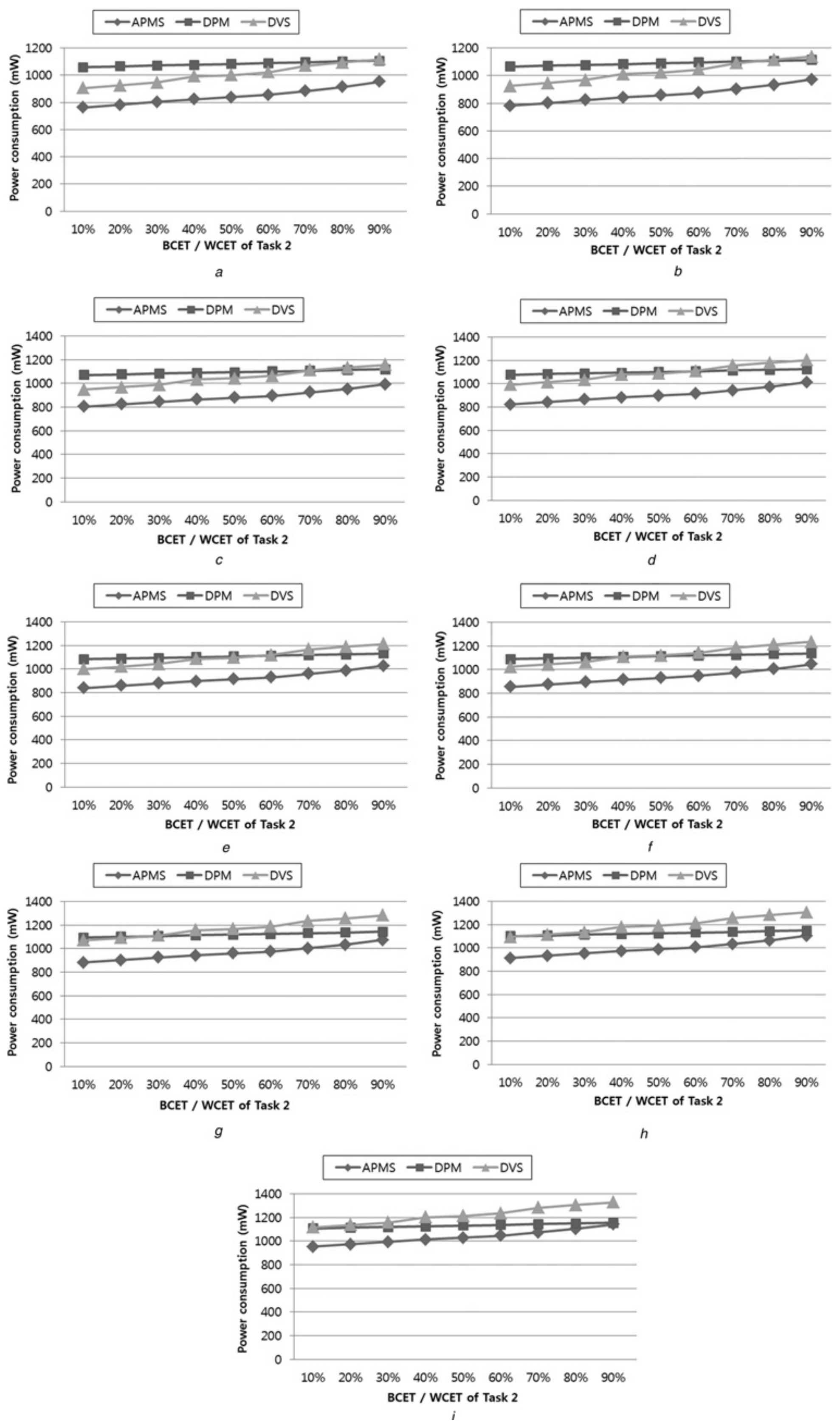

Fig. 5 *Power consumption of multiple tasks for the MP3's bit rate of 32 kb/s and different BECT/WCET ratios of Task 1 and Task 2*

a BECT/WCET of Task 1: 10%
b BECT/WCET of Task 1: 20%
c BECT/WCET of Task 1: 30%
d BECT/WCET of Task 1: 40%
e BECT/WCET of Task 1: 50%
f BECT/WCET of Task 1: 60%,
g BECT/WCET of Task 1: 70%
h BECT/WCET of Task 1: 80%
i BECT/WCET of Task 1: 90%

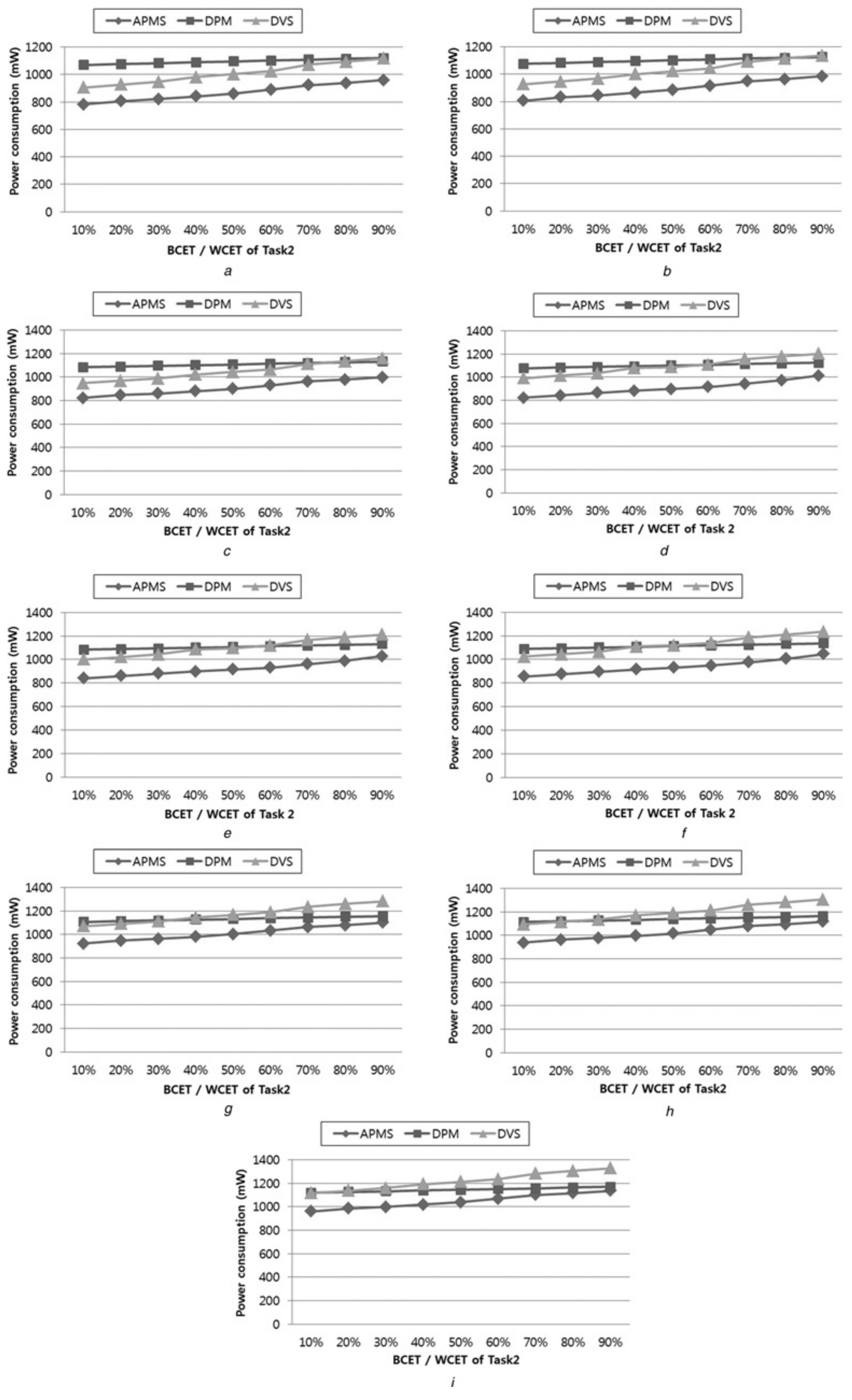

Fig. 6 *Power consumption of multiple tasks for the MP3's bit rate of 192 kb/s and different BECT/WCET ratios of Task 1 and Task 2*

a BECT/WCET of Task 1: 10%
b BECT/WCET of Task 1: 20%
c BECT/WCET of Task 1: 30%
d BECT/WCET of Task 1: 40%
e BECT/WCET of Task 1: 50%
f BECT/WCET of Task 1: 60%
g BECT/WCET of Task 1: 70%
h BECT/WCET of Task 1: 80%
i BECT/WCET of Task 1: 90%

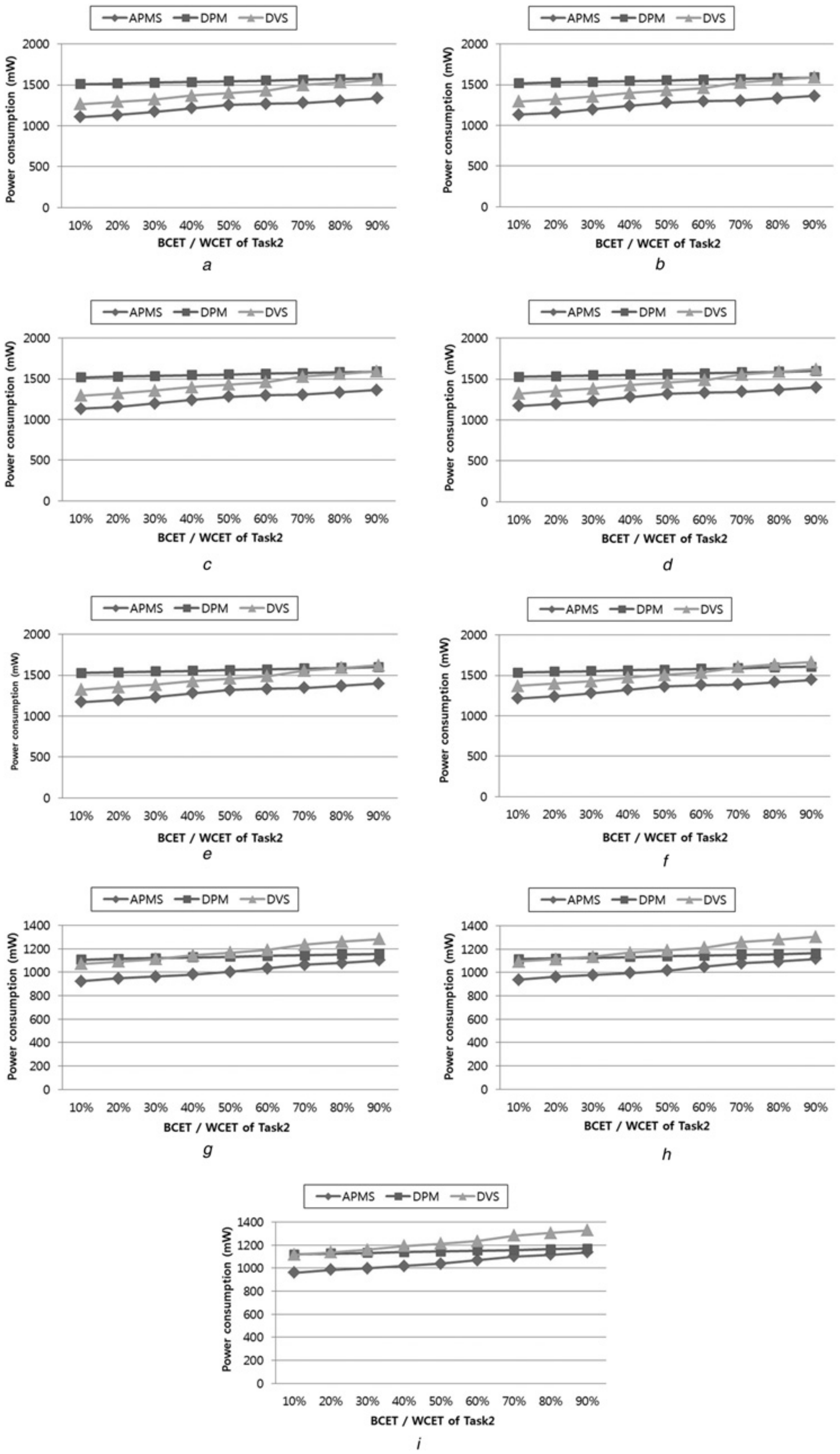

Fig. 7 *Power consumption of multiple tasks for the MP3's bit rate of 448 kb/s and different BECT/WCET ratios of Task 1 and Task 2*

a BECT/WCET of Task 1: 10%
b BECT/WCET of Task 1: 20%
c BECT/WCET of Task 1: 30%
d BECT/WCET of Task 1: 40%
e BECT/WCET of Task 1: 50%
f BECT/WCET of Task 1: 60%
g BECT/WCET of Task 1: 70%
h BECT/WCET of Task 1: 80%
i BECT/WCET of Task 1: 90%

This equation demonstrates that the BET depends on the period during which peripheral devices are in the 'run' state. Therefore if the execution time of a timeslot is longer than the BET, it is more efficient to change the state of a device into 'idle' in terms of power consumption. To conclude, as the execution time of the current timeslot becomes longer, since the state of idle devices which current timeslot does not use can be maintained long-term, power consumption can be reduced significantly.

4 Performance evaluation

We evaluate the performance of APMS through simulation. We use multimedia files (MP3) of which bit rates are between 32 and 448 kb/s. We assume that the processing times of substreams of multimedia data stream follow the uniform distribution of the ratio of the best-case execution time (BCET) to the WCET determined based on the maximum processing speed of PXA270 processor [17]. The processing times of substreams are determined by changing the BCETs in the range between 10 and 100% of the WCET so that we can measure performance using various processing times of substreams. The buffer size (2 MB) of APMS is the same as that of MPlayer for Linux. Finally, we compared power consumption of our system and naïve algorithms for DVS and DPM, since we shows adaptive selection between DVS and DPM is useful rather than clinging to one method.

Fig. 4 shows the amount of power consumed by single tasks for different bit rates: 32, 192 and 448 kb/s. According to our performance measurement, in the case of DPM, the amount of power consumption is affected only by the usage count of devices, but not by the ratios of BCET/WCET, since this mechanism is not applied to processor cores. In the case of DVS, when the BCET is short, that is, when the number of clock cycles required by a task is small, since the processor speed can be lowered significantly, the performance of DVS is better than that of DPM. However, as the BCET becomes longer, the number of clock cycles required by a task increases and the processor speed cannot be reduced significantly. Therefore the performance of DVS is lower than that of DPM, since DVS does not take into account the amount of power consumed by devices.

APMS has the lowest overall power consumption as the BCET approaches the lower bound of the WCET since it utilises both DPM and DVS simultaneously. On the other hand, as the BCET approaches the upper bound, it exhibits somewhat equal performance with DPM since it does not make sense to lower the processor speed to guarantee the deadlines required by multimedia streams. Our results show that power consumption of APMS was improved by 13.5 and 12.9% in average, respectively, when comparing it with those of DPM and DVS in a single-task environment.

Figs. 5–7 show power consumption of two tasks measured for different ratios of the BECT to the WCET from 10% to 90% and for three different bit rates (32, 192 and 448 kb/s) of an MP3 file. In the case of APMS, as shown in Figs. 5–7, power consumption can be reduced compared to the single-task case by scheduling timeslots of two tasks so that one or more peripheral devices can be used consecutively to avoid their frequent wakeups; however, when the BCET increases, it demonstrates identical performance with DPM since it is not possible to control processor voltage like the single-task case. According to our performance measurements, power consumption of APMS is reduced by 8.48 and 16.58%, respectively, in average when compared to DPM and DVS in a multiple-task environment.

5 Conclusions

In order to use mobile devices with limited resources over a long time, DPM and DVS have been widely used as power management methods. However, due to the nature of DPM, frequent changes in the states of peripheral devices may result in higher power consumption. In the case of DVS, although power consumption can be reduced by adjusting an acceptable quality of service range of the tasks, power consumption of peripheral devices was not considered. Thus, as the execution time of a task increased, power consumption of peripheral devices also increased, and consequently, the total power consumption also increased. Furthermore, even though low power consumption hardware has become more and more available, the relevant technologies have not been utilised yet properly at the level of software, including operating systems.

This paper proposed an APMS for devices that require low power consumption. It utilised the BTPS, the pattern analysis algorithm and useful features of both DPM and DVS. According to our performance measurements, when compared to DPM and DVS, the performance of APMS was better by 13.5 and 12.9%, respectively, in a single-task environment, and by 8.5 and 16.6%, respectively, in a multiple-task environment. Currently, we have been developing an enhanced version of EM algorithm in order to improve the correctness of pattern information delivered from the analysis module of APMS.

This paper proposed a system to deal with reducing power consumption by utilising pattern information for periodic multimedia task. We plan to investigate to deal with the pattern information for non-periodic tasks, since there exist many of non-periodic tasks in practice.

6 Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2010-0000414). This research is partially supported by Seoul National University of Science and Technology.

7 References

1 Hsu, R.C., Liu, C.T.: 'A reinforcement learning agent for dynamic power management in embedded systems', *J. Internet Technol.*, 2008, **9**, (4), pp. 347–353
2 Lai, W.K., Lai, H.C., Chen, J.C., Tsai, H.S.: 'Adjustable power control protocols in high load ad hoc wireless networks', *J. Internet Technol.*, 2007, **8**, (4), pp. 515–523
3 Shin, D., Kim, J.: 'Dynamic voltage scaling of mixed task sets in priority-driven systems', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2006, **25**, (3), pp. 438–453
4 Bang, S., Bang, K., Yoon, S., Chung, E.: 'Run-time adaptive workload estimation for dynamic voltage scaling', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2009, **28**, (9), pp. 1334–1347
5 Mao, J., Christos, C., Zhao, Q.: 'Optimal dynamic voltage scaling in energy-limited nonpreemptive systems with real-time constraints', *IEEE Trans. Mobile Comput.*, 2007, **6**, (6), pp. 678–688
6 Wang, W., Mishra, P.: 'PreDVS: preemptive dynamic voltage scaling for real-time systems using approximation scheme'. Proc. 47th Design Automation Conf., 2010, pp. 705–710
7 Lu, Y., Chung, E.Y., Simunic, T., Benini, L., Micheli, G.D.: 'Quantitative comparison of power management algorithms'. Proc. Design Automation and Test in Europe Conf. and Exhibition, Paris, France, March 2000, pp. 20–26

8 Sesic, A., Dautovic, S., Malbasa, V.: 'Dynamic power management of a system with a two-priority request queue using probabilistic-model checking', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2008, **27**, (2), pp. 403–407

9 Lee, S., Sakurai, T.: 'Runtime voltage hopping for low-power real-time systems'. Proc. Annual ACM IEEE Design Automation Conf., Los Angeles, California, USA, June 2000, pp. 806–809

10 Moon, T.K.: 'The expectation–maximization algorithm', *IEEE Signal Process. Mag.*, 1996, **13**, (6), pp. 47–60

11 Lu, Y., Benini, L., Micheli, G.D.: 'Low-power task scheduling for multiple devices'. Proc. Eighth Int. Workshop on Hardware/Software Codesign, San Diego, California, USA, May 2000, pp. 39–43

12 Shin, Y., Choi, K.: 'Power conscious fixed priority scheduling for hard real-time systems'. Proc. Design Automation Conf., New Orleans, Louisiana, USA, June 1999, pp. 134–139

13 Kim, W., Kim, J., Min, S.L.: 'A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis'. Proc. Design Automation and Test in Europe, Paris, France, March 2002, pp. 788–794

14 Lee, S., Sakurai, T.: 'Runtime voltage hopping for low-power real-time systems'. Proc. Design Automation Conf., Los Angeles, California, USA, June 2000, pp. 806–809

15 Cao, Z., Foo, B., He, L., Schaar, M.V.D.: 'Optimality and improvement of dynamic voltage scaling algorithms for multimedia applications', *IEEE Trans. Circuits Syst. I: Reg. Pap.*, 2010, **57**, (3), pp. 687–690

16 MPlayer Organization: 'The online documentation of MPlayer', http://www.mplayerhq.hu/DOCS/HTML-single/en/MPlayer.html, 2009

17 Intel Corperation: 'Intel PXA270 processor electrical, mechanical, and thermal specification', http://www.phytec.com/pdf/datasheets/PXA270_DS.pdf, 2005