Journal of Systems Architecture 59 (2013) 372–375

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Limited carry-in technique for real-time multi-core scheduling

Jinkyu Lee [a], Insik Shin [b,*]

[a] Dept. of Electrical Engineering and Computer Science, University of Michigan, USA
[b] Dept. of Computer Science, KAIST, South Korea

ARTICLE INFO

Article history:
Received 11 March 2013
Accepted 22 May 2013
Available online 6 June 2013

Keywords:
Real-time systems
Schedulability analysis
Limited carry-in technique
Real-time multi-core scheduling

ABSTRACT

Schedulability analysis has been widely studied to provide offline timing guarantees for a set of real-time tasks. The so-called *limited carry-in* technique, which can be orthogonally incorporated into many different multi-core schedulability analysis methods, was originally introduced for Earliest Deadline First (EDF) scheduling to derive a tighter bound on the amount of interference of carry-in jobs at the expense of investigating a pseudo-polynomial number of intervals. This technique has been later adapted for Fixed-Priority (FP) scheduling to obtain the carry-in bound efficiently by examining only one interval, leading to a significant improvement in multi-core schedulability analysis. However, such a successful result has not yet been transferred to any other non-FP scheduling algorithms. Motivated by this, this paper presents a generic limited carry-in technique that is applicable to any work-conserving algorithms. Specifically, this paper derives a carry-in bound in an algorithm-independent manner and demonstrates how to apply the bound to existing non-FP schedulability analysis methods for better schedulability.

1. Introduction

Schedulability analysis determines whether a set of real-time tasks can meet any given timing constraints (i.e., deadlines) under a certain scheduling algorithm on a specific computing platform. A substantial number of studies have been made to analyze schedulability in multi-core scheduling, introducing some successful results such as optimal scheduling algorithms for certain types of tasks (e.g., periodic tasks with implicit-deadlines). However, it yet lacks full understanding of how tasks behave on multi-cores under a given algorithm, particularly, for more general task types (e.g., periodic/sporadic tasks with constrained-deadlines). This leads to the development of many sufficient (but not exact) schedulability analysis methods [1–3].

Many different analysis methods share the perspective of investigating *interference* – how long the execution of a job of interest can be delayed due to the execution of other higher-priority jobs. Different analysis methods differ in how to derive a tight bound on the amount of such interference delay. Critical to this is how to tightly compute the contribution of *carry-in* jobs to the interference. A job is said to be a *carry-in* job for a given interval, if the job is released before the interval and does not yet finish its execution at the beginning of the interval.

A few studies have been made to compute the contribution of carry-in jobs. Baruah [2] introduced the so-called *limited carry-in* technique for Earliest Deadline First (EDF) [4] utilizing the concept of busy intervals. This technique requires to examine a pseudo-polynomial number of intervals for a tight carry-in bound. Guan et al. suggested to obtain a carry-in bound efficiently by examining only one interval under Fixed-Priority (FP) scheduling [4], taking advantage of the critical instant of a job (that maximizes its response time) under FP scheduling [5]. This technique yields reducing the carry-in bound substantially, resulting in a significant improvement in schedulability analysis for FP. A recent study [6] showed that such an FP-specific limited carry-in technique is applicable to a subset of tasks even under a dynamic-priority scheduling algorithm called Smallest Pseudo-Deadline First (SPDF), when the subset of tasks exhibits FP-oriented priority relationships between each other.

However, such an effective (in schedulability improvement) and efficient (in time-complexity) limited carry-in technique has not been utilized for other non-FP scheduling algorithms. Therefore, this paper seeks to develop a generic limited carry-in technique that can be applied to any work-conserving algorithms, under which any core is not idle as long as there is an unfinished job. To this end, we first recapitulate existing schedulability analysis techniques (in Section 2). Then, we develop a new carry-in bound, which can be applied to any work-conserving algorithm (in Section 2). Finally, we show applications how the new bound can be incorporated into existing schedulability analyses for better schedulability (in Section 3).

System Model. In this paper, we focus on a sporadic task model [7], in which a task τ_i in a task set τ is specified by the minimum separation T_i, the worst-case execution time C_i, and the relative

* Corresponding author. Tel.: +82 42 350 3524.
E-mail addresses: jinkyul@eecs.umich.edu (J. Lee), insik.shin@cs.kaist.ac.kr (I. Shin).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.05.012

deadline D_i. We focus on constrained deadline tasks, i.e., $C_i \leqslant D_i \leqslant T_i$. A task τ_i invokes a series of jobs, each separated from its predecessor by at least T_i time units, and supposed to finish its execution with D_i time units. Each job cannot be executed in parallel.

Also, we consider a multi-core system consisting of m identical cores, and preemptive work-conserving global algorithms, under which a higher-priority job can preempt a lower-priority job any time (preemptive), any core cannot be idle if there is an unfinished job (work-conserving), and a job can migrate from one core to another (global). Without loss of generality, let one time unit denote the quantum length, and all task parameters are assumed to be specified as multiples of the quantum length.

2. Limited carry-in technique for any work-conserving algorithm

Many schedulability analysis methods employ the concept of *interference*. Let $I_{k\leftarrow i}(a,b)$ denote the interference of τ_i on τ_k in $[a,b)$, meaning the cumulative length of all intervals in $[a,b)$ such that a job of τ_i executes but the job of τ_k of interest cannot although it is ready to execute.

Then, considering that a job of τ_k cannot be executed in a time slot, only when m other jobs are executed, the interference-based schedulability analysis framework has been developed in [1,3], as follows.

Lemma 1. *(Theorem 3 in [1], Theorem 6 in [3]) Every job of a task $\tau_k \in \tau$ finishes its execution within ℓ time units if the following inequality holds for $\ell \leqslant D_k$:*

$$C_k + \left\lfloor \frac{1}{m} \cdot \max_{t\in\mathbf{T}} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\left(I_{k\leftarrow i}(t, t+\ell), \ell - C_k + 1\right) \right\rfloor \leqslant \ell, \tag{1}$$

where $\mathbf{T} \triangleq \{t |$the release time of a job of $\tau_k\}$. Then, if Eq. (1) holds for all $\tau_k \in \tau$, we deem τ is schedulable.

It is difficult to calculate the exact value of $I_{k\leftarrow i}(t, t+\ell)$ offline, because the value depends not only on the scheduling algorithm, but also on job release patterns before t. Hence, existing schedulability analysis techniques have derived two types of upper-bounds on $I_{k\leftarrow i}(t, t+\ell)$: (i) algorithm-independent ones and (ii) algorithm-specific ones.

Regarding (i), a release (and execution) pattern that maximizes the execution of jobs of τ_i has been identified [1]. As shown in Fig. 1(a), this pattern allows a task τ_i to have a carry-in job in the interval of interest. In the figure, the first (carry-in) job executes as late as possible, and the following jobs execute as early as possible. Here S_i means the slack value of τ_i, a difference between the finishing time and deadline of a job of τ_i; it has been detailed in [1,3] how to calculate S_i with existing schedulability analyses. Then, the amount of the maximum execution of jobs of τ_i in an interval of length ℓ is calculated by [1,3]

$$W_i^{\mathrm{CI}}(\ell) \triangleq N_i^{\mathrm{CI}}(\ell) \cdot C_i + \min\left(C_i, \ell + D_i - S_i - C_i - N_i^{\mathrm{CI}}(\ell) \cdot T_i\right), \tag{2}$$

where $N_i^{\mathrm{CI}}(\ell)$ denotes the number of jobs of τ_i whose deadlines are within an interval of length ℓ (e.g., the first two jobs in Fig. 1(a)), and calculated by $\left\lfloor \frac{\ell + D_i - S_i - C_i}{T_i} \right\rfloor$.

Since a job can interfere with another job only when it is executed, $I_{k\leftarrow i}(t, t+\ell) \leqslant W_i^{\mathrm{CI}}(\ell)$ holds for all $t, \ell \geqslant 0$ and $\tau_i \in \tau$. Then, Eq. (1) with replacing $I_{k\leftarrow i}(t, t+\ell)$ by $W_i^{\mathrm{CI}}(\ell)$ yields a safe schedulability analysis of any work-conserving algorithm [1,3], and this schedulability analysis assumes that all tasks can have carry-in jobs.

In case that a task τ_i cannot have any carry-in job in an interval of length ℓ, Fig. 1(b) illustrates a release and (execution) pattern that maximizes the execution of jobs of τ_i. Here the number of jobs of τ_i whose release time and deadline are in the interval is $N_i^{\mathrm{NC}}(\ell) \triangleq \left\lfloor \frac{\ell}{T_i} \right\rfloor$, and the amount of the maximum execution of jobs of τ_i is calculated by [8,5]

$$W_i^{\mathrm{NC}}(\ell) \triangleq N_i^{\mathrm{NC}}(\ell) \cdot C_i + \min\left(C_i, \ell - N_i^{\mathrm{NC}}(\ell) \cdot T_i\right). \tag{3}$$

We can easily check $W_i^{\mathrm{NC}}(\ell) \leqslant W_i^{\mathrm{CI}}(\ell)$ for all $\ell \geqslant 0$ and $\tau_i \in \tau$.

Since we do not know offline whether a task has its carry-in job in the interval of interest, $W_i^{\mathrm{NC}}(\ell)$ cannot be a safe upper-bound of $I_{k\leftarrow i}(t, t+\ell)$. Instead, $I_{k\leftarrow i}(t, t+\ell) \leqslant W_i^{\mathrm{NC}}(\ell)$ holds only if τ_i does not have any carry-in job in $[t, t+\ell)$. However, we can derive a safe upper-bound on the amount of execution with limited carry-in jobs, as stated in the following lemma.

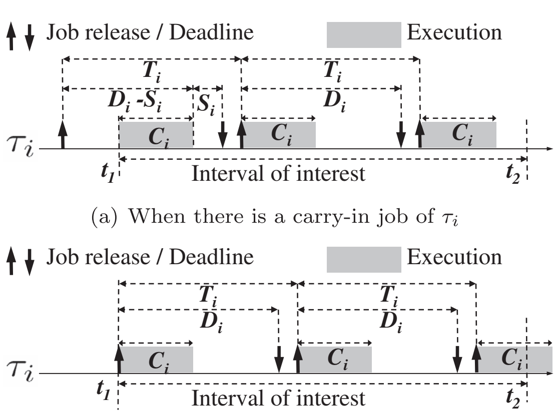
Lemma 2. *Let Γ denote a set of intervals (not necessarily continuous) of length $\ell - \alpha$ ($\alpha \geqslant 0$) over $[t, t+\ell)$. Suppose there are at most $m-1$ tasks which have their carry-in jobs in $[t, t+\ell)$. Then, the amount of execution of jobs of tasks in τ in any Γ is upper-bounded by $\mathscr{F}(\ell, \alpha)$, where*

$$\mathscr{F}(\ell, \alpha) \triangleq \sum_{\tau_i \in \tau} \min\left(W_i^{\mathrm{NC}}(\ell), \ell - \alpha\right) + \sum_{m-1 \text{ largest } \tau_i \in \tau} \min\left(W_i^{\mathrm{CI}}(\ell), \ell - \alpha\right) - \min\left(W_i^{\mathrm{NC}}(\ell), \ell - \alpha\right). \tag{4}$$

Proof. Recall that $W_i^{\mathrm{CI}}(\ell)$ and $W_i^{\mathrm{NC}}(\ell)$ represent the maximum amount of execution of jobs of τ_i in an interval of length ℓ, respectively when there is a carry-in job and no carry-in job of τ_i in the interval. Considering the length of Γ is $\ell - \alpha$, $\min(W_i^{\mathrm{CI}}(\ell), \ell - \alpha)$ (likewise $\min(W_i^{\mathrm{NC}}(\ell), \ell - \alpha)$) is an upper-bound on the maximum amount of execution of jobs of τ_i in Γ in case of the existence of a carry-in job of τ_i (likewise no carry-in job of τ_i).

Since we do not know which task has its carry-in job, we initially add $\min\left(W_i^{\mathrm{NC}}(\ell), \ell - \alpha\right)$ for all tasks in τ. Then, we add the difference between $\min\left(W_i^{\mathrm{CI}}(\ell), \ell - \alpha\right)$ and $\min\left(W_i^{\mathrm{NC}}(\ell), \ell - \alpha\right)$ for $m-1$ tasks with the largest difference. Then, for any combination of at most $m-1$ tasks with carry-in jobs, $\mathscr{F}(\ell, \alpha)$ in Eq. (4) is a safe upper-bound on the amount of execution of jobs of tasks in τ in any Γ, provided that there are at most $m-1$ tasks with carry-in jobs in $[t, t+\ell)$. $\square$

Using the lemma, we finally present the main theorem of this paper.



(a) When there is a carry-in job of τ_i

(b) When there is no carry-in job of τ_i

Fig. 1. A release pattern that maximizes the amount of execution of jobs of τ_i in an interval starting t_1.

Theorem 1. *Under any work-conserving algorithm, the total amount of interference in Eq. (1) is upper-bounded as follows:*

$$\max_{t\in\mathbf{T}} \sum_{\tau_i\in\tau-\{\tau_k\}} \min\left(I_{k\leftarrow i}(t,t+\ell),\ell-C_k+1\right) \text{ in Eq. (1)} \leqslant \mathscr{F}(\ell,C_k-1) \text{ in Eq. (4)}. \quad (5)$$

Proof. Suppose that Eq. (1) with replacing the LHS of Eq. (5) by the RHS satisfies the following inequality for $\ell \leqslant D_k$:

$$C_k+\left\lfloor \frac{1}{m}\cdot(\mathscr{F}(\ell,C_k-1) \text{ in Eq. (1)})\right\rfloor \leqslant \ell \iff (\mathscr{F}(\ell,C_k-1) \text{ in Eq. (4)}) < m\cdot(\ell-C_k+1). \quad (6)$$

Then, we will prove that any job of τ_k finishes its execution within ℓ time units. Without loss of generality, we focus on a job of τ_k whose release time and deadline are t_0 and t_0+D_k.

We choose a time instant t_0-x ($x\geqslant 0$) such that an interval $[t_0-x,t_0)$ is the maximum busy interval, in which all m cores are occupied in $[t_0-x,t_0)$, but at least one core is idle in $[t_0-x-1,t_0-x)$. Then, by definition, there are at most $m-1$ tasks which have their carry-in jobs in an interval starting from t_0-x.

We consider two cases: (i) $0\leqslant x<\ell-C_k+1$ and (ii) $x\geqslant \ell-C_k+1$.

In case of (i), we focus on $[t_0-x,t_0)\cup\Gamma'$, where Γ' denotes a set of intervals (not necessarily continuous) of length $\ell-C_k+1-x$ over $[t_0,t_0-x+\ell)$; Γ' is chosen such that the amount of execution of jobs of tasks in τ in Γ' is maximized. If we apply Lemma 2 with $t=t_0-x$ and $\Gamma=[t_0-x,t_0)\cup\Gamma'$, Lemma 2 and Eq. (6) imply that the amount of execution of jobs of tasks in τ in $[t_0-x,t_0)\cup\Gamma'$ is strictly less than $m\cdot(\ell-C_k+1)$. Since there are $m\cdot x$ execution of jobs of tasks in τ in $[t_0-x,t_0)$ (because the interval is a busy interval), there are strictly less than $m\cdot(\ell-C_k+1-x)$ executions in Γ', which means at least one time slot in Γ' is a non-busy slot in which at least one core is idle. Then, by the definition of Γ', each time slot in $[t_0,t_0-x+\ell)\setminus\Gamma'$ is non-busy. Therefore, the job of interest can be executed in all time slots in $[t_0,t_0-x+\ell)\setminus\Gamma'$ (whose length is C_k-1) and at least one slot in Γ', which means the job of interest finishes C_k amount of execution in $[t_0,t_0-x+\ell)$.

In case of (ii), we focus on $[t_0-x,t_0-x+\ell-C_k+1)$. We apply Lemma 2 with $t=t_0-x$ and $\Gamma=[t_0-x,t_0-x+\ell-C_k+1)$. Then, Lemma 2 and Eq. (6) imply that the amount of execution of jobs of tasks in τ in $[t_0-x,t_0-x+\ell-C_k+1)$ is strictly less than $m\cdot(\ell-C_k+1)$. By the definition of $[t_0-x,t_0)$, $[t_0-x,t_0-x+\ell-C_k+1)$ is a busy interval, and therefore the amount should be exactly as much as $m\cdot(\ell-C_k+1)$. Therefore, if Eq. (6) holds, $[t_0-x,t_0-x+\ell-C_k+1)$ cannot be a busy interval, which is a contradiction.

3. Applications of the limited carry-in technique

While Theorem 1 can be combined with most (if not all) interference-based schedulability analyses, we now present examples of the application.

In [3], a schedulability analysis for any work-conserving algorithm has been developed, by using $W_i^{\mathrm{CI}}(\ell)$ as an upper-bound of $I_{k\leftarrow i}(t,t+\ell)$ as follows:

$$\max_{t\in\mathbf{T}} \sum_{\tau_i\in\tau-\{\tau_k\}} \min\left(I_{k\leftarrow i}(t,t+\ell),\ell-C_k+1\right) \text{ in Eq. (1)} \leqslant \sum_{\tau_i\in\tau-\{\tau_k\}} \min\left(W_i^{\mathrm{CI}}(\ell),\ell-C_k+1\right). \quad (7)$$

In [1], a schedulability analysis for a given scheduling algorithm has been developed, by deriving an algorithm-specific upper-bound on $I_{k\leftarrow i}(t,t+\ell)$ as follows:

$$\max_{t\in\mathbf{T}} \sum_{\tau_i\in\tau-\{\tau_k\}} \min\left(I_{k\leftarrow i}(t,t+\ell),\ell-C_k+1\right) \text{ in Eq. (1)} \leqslant \sum_{\tau_i\in\tau-\{\tau_k\}} \min\left(W_i^{\mathrm{CI}}(\ell),E_{k\leftarrow i},\ell-C_k+1\right), \quad (8)$$

where $E_{k\leftarrow i}$ is an algorithm-specific upper-bound on $I_{k\leftarrow i}(t,t+\ell)$ for any $\ell\leqslant D_k$. For example, under EDF and Earliest Deadline first until Zero-Laxity (EDZL) [9], $E_{k\leftarrow i}$ is calculated by $N_i^{\mathrm{NC}}(D_k)\cdot C_i+\min\left(C_i,\max(0,D_k-N_i^{\mathrm{NC}}(D_k)\cdot T_i-S_i)\right)$ [1,10], considering a job with later deadline cannot interfere with another job with earlier deadline under EDF, and the former can interfere with the latter only when the former enters the zero-laxity state under EDZL.[1]

If we apply Theorem 1, we can derive tighter upper-bounds than the ones presented in Eqs. (7) and (8). For the schedulability analysis for any work-conserving algorithm, we use the following upper-bound:

$$\text{LHS of Eq. (7)} \leqslant \min(\text{RHS of Eq. (7)},\ \text{RHS of Eq. (5)}). \quad (9)$$

Similarly, for the schedulability analysis for a given scheduling algorithm, we use the following upper-bound:

$$\text{LHS of Eq. (8)} \leqslant \min(\text{RHS of Eq. (8)},\ \text{RHS of Eq. (5)}). \quad (10)$$

Here we present two task sets which are not schedulable with the upper-bound in the RHS of Eq. (7) or (8), but schedulable with the upper-bound in the RHS of Eq. (9) or (10).

Example 1. Consider a set of three tasks $\tau=\{\tau_1(T_1=4, C_1=1,D_1=4),\tau_2(4,2,4),\tau_3(4,2,4)\}$ on a two-core platform, and its schedulability is tested by the state-of-the-art schedulability analysis framework, i.e., response-time analysis [1]. Using the RHS of Eq. (7), we cannot guarantee that τ_1 finishes its execution within 4 time units under any work-conserving algorithm. However, with the RHS of Eq. (9), we can guarantee that all three tasks meet their deadlines under any work-conserving algorithm.

Example 2. Consider a set of four tasks $\tau=\{\tau_1(T_1=2,C_1=1, D_1=2),\tau_2(2,1,2),\tau_3(7,3,7),\tau_4(10,1,10)\}$ on a two-core platform, with the same schedulability analysis framework as Example 1. Using the RHS of Eq. (8), we cannot guarantee that τ_1 and τ_2 finish their execution within 2 time units under EDF [1]; note that the limited carry-in technique in [2] also cannot guarantee the schedulability of τ. However, with the RHS of Eq. (10), we can guarantee that all four tasks do not miss their deadlines under EDF. The same holds for EDZL; the state-of-the-art EDZL test in [10] cannot guarantee that τ is schedulable, but we can do with the RHS of Eq. (10).

As shown in the examples, Theorem 1, once incorporated into existing schedulability analyses (e.g., Eqs. (9)), covers additional schedulable task sets which are not deemed schedulable by the existing ones.

4. Conclusion

In this paper, we derived a new upper-bound on the amount of execution under any work-conserving algorithm by limiting carry-in jobs, and demonstrated how to utilize this upper-bound to improve existing schedulability analyses. Extending the idea of the limited carry-in, it would be interesting to derive tighter,

[1] Due to the necessary condition for jobs to enter the zero-laxity state, $E_{k\leftarrow i}$ is also an upper-bound under EDZL; details are given in [10].

algorithm-specific upper-bounds. In particular, since Theorem 1 can be applied even to work-conserving non-preemptive scheduling, we expect to improve the state-of-the-art schedulability analyses for non-preemptive scheduling [11–14]. Another direction of future work is to explore other properties of our limited carry-in techniques (e.g., sustainability [15]) and extend the technique towards other systems (e.g., time-triggered embedded systems [16]).

Acknowledgment

This work was supported in part by BRL (2009-0086964), BSRP (2010-0006650, 2012-R1A1A1014930), NCRC (2012-0000980), NIPA (H0503-12-1041), KEIT (2011-10041313), DGIST CPS Global Center, and KIAT (M002300089) funded by the Korea Government (MEST/MKE).

References

[1] M. Bertogna, M. Cirinei, Response-time analysis for globally scheduled symmetric multiprocessor platforms, in: Proceedings of IEEE Real-Time Systems Symposium, 2007, pp. 149–160.
[2] S. Baruah, Techniques for multiprocessor global schedulability analysis, in: Proceedings of IEEE Real-Time Systems Symposium, 2007, pp. 119–128.
[3] M. Bertogna, M. Cirinei, G. Lipari, Schedulability analysis of global scheduling algorithms on multiprocessor platforms, IEEE Transactions on Parallel and Distributed Systems 20 (2009) 553–566.
[4] C. Liu, J. Layland, Scheduling algorithms for multi-programming in a hard-real-time environment, Journal of the ACM 20 (1) (1973) 46–61.
[5] R. Davis, A. Burns, Improved priority assignment for global fixed priority preemptive scheduling in multiprocessor real-time systems, Real-Time Systems 47 (2011) 1–40.
[6] H.S. Chwa, H. Back, S. Chen, J. Lee, A. Easwaran, I. Shin, I. Lee, Extending task-level to job-level fixed priority assignment and schedulability analysis using pseudo-deadlines, in: Proceedings of IEEE Real-Time Systems Symposium, 2012, pp. 51–62.
[7] A. Mok, Fundamental design problems of distributed systems for the hard-real-time environment, Ph.D. Thesis, Massachusetts Institute of Technology, 1983.
[8] N. Guan, M. Stigge, W. Yi, G. Yu, New response time bounds for fixed priority multiprocessor scheduling, in: Proceedings of IEEE Real-Time Systems Symposium, 2009, pp. 387–397.
[9] S.K. Lee, On-line multiprocessor scheduling algorithms for real-time tasks, in: IEEE Region 10's Ninth Annual International Conference, 1994, pp. 607–611.
[10] T.P. Baker, M. Cirinei, M. Bertogna, EDZL scheduling analysis, Real-Time Systems 40 (2008) 264–289.
[11] N. Guan, W. Yi, Z. Gu, Q. Deng, G. Yu, New schedulability test conditions for non-preemptive scheduling on multiprocessor platforms, in: Proceedings of IEEE Real-Time Systems Symposium, 2008, pp. 137–146.
[12] H. Leontyev, J.H. Anderson, A unified hard/soft real-time schedulability test for global EDF multiprocessor scheduling, in: Proceedings of IEEE Real-Time Systems Symposium, 2008, pp. 375–384.
[13] N. Guan, W. Yi, Q. Deng, Z. Gu, G. Yu, Schedulability analysis for non-preemptive fixed-priority multiprocessor scheduling, Journal of Systems Architecture 57 (5) (2011) 536–546.
[14] J. Lee, K.G. Shin, Controlling preemption for better schedulability in multi-core systems, in: Proceedings of IEEE Real-Time Systems Symposium, 2012, pp. 29–38.
[15] A. Burns, S. Baruah, Sustainability in real-time scheduling, Journal of Computing Science and Engineering 2 (1) (2008) 74–97.
[16] H. Kopetz, On the design of distributed time-triggered embedded systems, Journal of Computing Science and Engineering 2 (4) (2008) 340–356.

Jinkyu Lee received B.S., M.S. and Ph.D. degrees in Computer Science in 2004, 2006 and 2011, respectively, from Korea Advanced Institute of Science and Technology (KAIST), South Korea. Since October 2011, he is a research fellow/visiting scholar in Department of Electrical Engineering and Computer Science, University of Michigan, USA. His research interests include system design and analysis with timing guarantees, QoS support, and resource management in real-time embedded systems and cyber-physical systems. He won the best student paper award from the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2011, and the best paper award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.

Insik Shin is currently an associate professor in Dept. of Computer Science at KAIST, South Korea, where he joined in 2008. He received a B.S. from Korea University, an M.S. from Stanford University, and a Ph.D. from University of Pennsylvania all in Computer Science in 1994, 1998, and 2006, respectively. He has been a post-doctoral research fellow at Malardalen University, Sweden, and a visiting scholar at University of Illinois, Urbana-Champaign until 2008. His research interests lie in cyber-physical systems and real-time embedded systems. He is currently a member of Editorial Boards of Journal of Computing Science and Engineering. He has been co-chairs of various workshops including satellite workshops of RTSS, CPS-Week and RTCSA and has served various program committees in real-time embedded systems, including RTSS, RTAS, ECRTS, and EMSOFT. He received best paper awards, including Best Paper Awards from RTSS in 2003 and 2012, Best Student Paper Award from RTAS in 2011, and Best Paper runner-ups at ECRTS and RTSS in 2008.