Volume 32, March 2014
ISSN 0167-739X
ELSEVIER
THE INTERNATIONAL JOURNAL OF
FGCS
FUTURE GENERATION COMPUTER SYSTEMS
GRID COMPUTING AND eSCIENCE
Editor-in-Chief:
Peter Sloot
Associate Editors:
David Abramson
Ilkay Altintas
Available online at www.sciencedirect.com
ScienceDirect

Future Generation Computer Systems 32 (2014) 253–259

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Effective real-time scheduling algorithm for cyber physical systems society

Sanghyuk Park[a], Jai-Hoon Kim[b,*], Geoffrey Fox[c]

[a] Department of Network Centric Warfare Engineering, Ajou University, South Korea
[b] Graduate School of Information and Communication, Ajou University, South Korea
[c] Pervasive Technology Institute, Indiana University, Bloomington, IN, USA

HIGHLIGHTS

- Computers and physical systems are tightly coupled in cyber physical society.
- Conventional systems only consider cyber space.
- CPS should also consider physical, socio and mental space.
- Proposed scheduling algorithm considering physical factors.
- Efficiency of algorithm is verified by mathematical analysis and simulation.

ARTICLE INFO

Article history:
Received 4 July 2012
Received in revised form
29 September 2013
Accepted 4 October 2013
Available online 12 October 2013

Keywords:
CPS (Cyber Physical System)
Real-time scheduling algorithm
LST (Least Slack Time First)
ELST (Effective Least Slack Time First)
H-ELST (Heuristic Effective Least Slack Time First)

ABSTRACT

CPS (Cyber Physical Systems) tightly couple their cyber factor and physical factor in distributed computing or Grids environments to provide real-time services such as avionics, transportation, manufacturing processes, energy, healthcare, etc. We need to consider not only the cyber space (CPU, network, storage systems, etc.) and the physical space (location, migration, etc.) but also the socio space and mental space for the precise analysis and useful services. In this paper, real-time scheduling algorithms, namely ELST (Effective Least Slack Time First) and H-ELST (Heuristic-Effective Least Slack Time First), are presented for CPS, where servicing node needs to move to serviced node for real-time services. We measure the real-time performance in terms of deadline meet ratio by mathematical analysis and simulations. The results show that our algorithms reduce a deadline miss ratio approximately up to 50% and 20% compared to the conventional real-time scheduling algorithm, FIFO (First In First Out) and LST (Least Slack Time First), respectively.

1. Introduction

Timing issues are critical in real-time systems such as robot control [1,2], NCO (Network Centric Operations) systems [3–6], flight control, on-line multimedia systems [7], and real-time stock trading system, etc. [8–10]. Many real-time scheduling algorithms such as RM (rate monotonic) [11,12], EDF (earliest deadline first) [12–14], and LST (least slack time first) [12,14] deal with resource (CPU and network bandwidth) scheduling to maximize real-time performance (e.g., deadline meet ratio) [7,14]. As CPS (cyber physical system [15–18] and cyber physical society [19–21]) such as avionics, transportation, manufacturing processes, energy, healthcare, in which computers and physical systems (also, society and mental) are tightly coupled and timing is critical, is fast growing, real-time scheduling for CPS becomes the new research issues in the real-time systems [22,23].

In other aspects, as real-time applications become complex and relevant tasks and resources are widely distributed, we have to study the real-time scheduling in distributed computing infrastructures and Grids. For examples, in Grids infrastructures (e.g., EGI (European Grid Infrastructure) [24], SEE-GRID (South Eastern European Grid-enabled e-Infrastructure) [25], and EELA (E-science grid facility for Europe and Latin America) [26]) many tasks concurrently request various types of distributed resources. Middleware has to coordinate the resource allocation to provide services and guarantee a SLA. In these distributed environments, the real-time scheduling must consider transfer delays as task and data migrations among nodes having computing resources are common. Red Hat Enterprise MRG (Messaging, Real time, and Grid) Real time [27]

* Corresponding author. Tel.: +82 31 219 2546; fax: +82 31 219 1614.
E-mail address: jaikim@ajou.ac.kr (J.-H. Kim).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.10.003

Table 1
Real-time scheduling for CPS.

	Conventional real-time scheduling	Real-time scheduling for CPS
Scheduling resources	CPU, BW, memory, I/O	Servicing node
Scheduling environment	Cyber environment	Cyber and physical environment
Scheduling parameters	Cyber factors (Period, execution time, release time, deadline, etc.)	Cyber factors and physical factors (period, execution time, release time, deadline, migration delay time, etc.)
Migration	No migration is required for CPU and Job. * We consider CPU is serving node and Job is serviced node	Migration is required for CPU and Job.
Well-known Scheduling algorithm	RM, EDF, LST, etc.	None
Spatial issues	Do not consider spatial issues	Physical migration delay time (between servicing node to serviced node)
Considering issues	Execution time, release time, deadline, laxity	Execution time, release time, effective deadline (deadline−moving time), effective laxity (laxity−moving delay time)

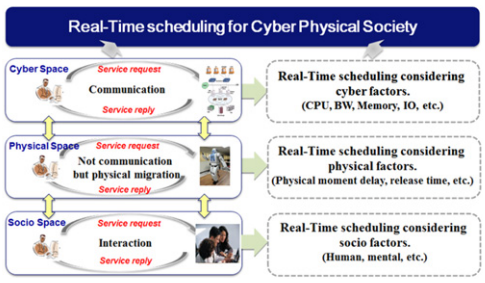

Real-Time scheduling for Cyber Physical Society
Cyber Space
Service request
Communication
Service reply
Real-Time scheduling considering cyber factors. (CPU, BW, Memory, IO, etc.)
Physical Space
Service request
Not communication but physical migration
Service reply
Real-Time scheduling considering physical factors. (Physical moment delay, release time, etc.)
Socio Space
Service request
Interaction
Service reply
Real-Time scheduling considering socio factors. (Human, mental, etc.)

Fig. 1. Real-time scheduling for CPS.

provides a high level of predictability for consistent low-latency response times to meet the requirement of time-sensitive workloads. Many large-scale distributed applications require real-time responses to meet soft deadlines. Ref. [28] design and implement the real-time volunteer computing platform called RT-BOINC to schedule the real-time task and execute on the volunteer resources.

Many real-time scheduling algorithms have been proposed and widely used [11,12,14]. However, in cyber physical systems society, we need to consider not only cyber space (CPU, network, storage systems, etc.) and physical space (location, migration, etc.) but also socio space and mental space [19–21]. Fig. 1 shows the real-time scheduling model for cyber physical systems society [19–21]. Effective release time and deadline of real-time tasks may be different depending on the location and physical migration delay time of nodes participating in CPS. Real-time scheduling algorithms have to be modified to include spatial factors. Conventional cyber real-time system schedules CPU or network bandwidth. However, in real-time scheduling for CPS, location is matter. Location of nodes in CPS affects the effective release time and deadline.

In this paper, we propose new real-time scheduling algorithms for CPS, where the servicing node needs to move to serviced node for real-time services. If we assume, for an example, there are many scattered customers requesting real-time services but only one servicing staff exists in the area, real-time scheduling is necessary to maximize the performance (e.g., deadline meet ratio). In this case, the conventional real-time scheduling algorithm is not proper because the real-time scheduling does not consider physical factors (e.g., locations of customer of servicing staff, migration delay between the locations, etc.). In CPS, the physical factors, however, are not entirely predictable or easy to change [10], leading to problems such as missed task deadlines, faults of cyber systems, and faults of physical systems [11,12]. Such problems are very serious in CPS and could cause widespread social upheaval, as well as huge inconvenience and economic loss for individuals and industry alike. We propose a method of solving such problems by introducing new real-time scheduling algorithms for CPS.

Real-time scheduling for CPS differs from conventional real-time scheduling in many aspects. Table 1 highlights the key differences between the conventional real-time scheduling and the CPS real-time scheduling. As in many kinds of CPS, where servicing nodes must move to the location to perform real-time services, time required for moving has to be included in the real-time scheduling. In some CPS cases, servicing node cannot move to serviced nodes. As a future work, we will consider another case where serviced nodes move to servicing node. Also, we will make real-time scheduling algorithms considering social factors mentioned in Refs. [19–21] such as socio space, mental space, etc.

The remainder of this paper is organized as follows. Section 2 presents the real-time scheduling algorithms for CPS. Our algorithms are evaluated in Section 3. Finally, we conclude the paper and point out the future works.

2. Real-time scheduling model in CPS

In this section, we propose a real-time scheduling for CPS and compare the real-time performance (deadline meet ratio) between the conventional real-time scheduling and the proposed real-time scheduling for CPS. We assume parameters for real-time systems as follows:

- l_i: slack (laxity) time of task i (exponential distribution of average $1/\lambda$)
- e_i: execution time of task i (evenly distributed on $[0, E]$)
- m_i: migration time of servicing(computing) node to task(serviced node) i (evenly distributed in $[0, M]$).

Deadline meet ratio (DM) of task A without confliction against other tasks is the probability of the slack time l_A being greater than the moving time m_A (servicing node moving to serviced node (task A) within slack time l_A). As distribution of l_A is $\lambda e^{-\lambda t}$, the deadline meet ratio of a task A (DM_A (λ, m)) is computed as follows:

$$\mathrm{DM}_A(\lambda, m) = \int_m^{\infty} \lambda e^{-\lambda t} dt = e^{-\lambda m}. \tag{1}$$

As m is assumed to evenly distributed $[0, M]$, an average deadline meet ratio is:

$$\mathrm{Mean}(\mathrm{DM}_A(\lambda, m)) = \frac{1}{M}\int_0^M e^{-\lambda m} dm = \frac{1}{\lambda M}(1 - e^{-\lambda M}). \tag{2}$$

For a simple demonstration, we compute a deadline meet ratio when two tasks conflict each other. (We also perform simulation in more realistic scenarios as described in Section 3.2.) We compute deadline meet ratios for three different scheduling algorithms: FIFO (First In First Service), LST (Least Slack Time First), ELST (Effective Least Slack Time First for CPS) scheduling algorithms.

2.1. FIFO (first in first out)

We assume that the task A arrived just before the other task B. A deadline meet ratio of task A is mean (DM_A (λ, m)) as task A is performed without confliction. As task B can be scheduled after task A, the deadline meet ratio of task B is the probability of the slack time of task B (l_B) being greater than $m_A + e_A + m_B$. Thus, the deadline meet ratio of task B following task A (Mean(DM_B (λ, m_A, e_A, m_B))) is computed as follows:

$$\begin{aligned}\mathrm{Mean}(\mathrm{DM}_B(\lambda, m_A, e_A, m_B)) \\ &= \frac{1}{M^2E}\int_0^M\int_0^E\int_0^M \mathrm{e}^{-\lambda(m_A+e_A+m_B)}dm_A de_A dm_B \\ &= \frac{(1-\mathrm{e}^{-\lambda M})^2(1-\mathrm{e}^{-\lambda E})}{\lambda^3 M^2 E}. \end{aligned} \quad (3)$$

Now, we obtain the deadline meet ratio of FIFO scheduling algorithm when task A and task B conflict.

$$\begin{aligned}\mathrm{DM}_{\mathrm{fifo}} &= \frac{\{\mathrm{Mean}(\mathrm{DM}_A(\lambda, m)) + \mathrm{Mean}(\mathrm{DM}_B(\lambda, m_A, e_A, m_B))\}}{2} \\ &= \left\{\frac{1}{\lambda M}(1-\mathrm{e}^{-\lambda M}) + \frac{(1-\mathrm{e}^{-\lambda M})^2(1-\mathrm{e}^{-\lambda E})}{\lambda^3M^2E}\right\}\Big/2. \end{aligned} \quad (4)$$

2.2. LST (least slack time first)

When task A and task B conflict, a task with least slack time is scheduled first. When we assume that the slack time of task A is shorter than that of task B, the slack time of task A is the exponential distribution ($2\lambda\mathrm{e}^{-2\lambda t}$) of average $1/(2\lambda)$ while the slack of task B is the exponential distribution of $2\lambda\mathrm{e}^{-\lambda t} - 2\lambda\mathrm{e}^{-2\lambda t} = 2\lambda\mathrm{e}^{-\lambda t}(1-\lambda\mathrm{e}^{-\lambda t})$. (We can obtain it by using the Markov model.) A deadline meet ratio of task A is mean (DM_A (2λ, m)) as task A is performed without confliction.

As task B of longer slack time can be scheduled after task A of shorter slack time, the deadline meet ratio of task B is the probability of the slack time of task B (l_B) being greater than $m_A + e_A + m_B$. Thus, an average deadline meet ratio of task B following task A is computed as follows:

$$\begin{aligned}&\frac{1}{M^2E}\int_0^M\int_0^E\int_0^M (2\mathrm{e}^{-\lambda(m_A+e_A+m_B)} \\ &\quad - \mathrm{e}^{-2\lambda(m_A+e_A+m_B)})dm_A de_A dm_B \\ &= \frac{2(1-\mathrm{e}^{-\lambda M})^2(1-\mathrm{e}^{-\lambda E})}{\lambda^3M^2E} - \frac{(1-\mathrm{e}^{-2\lambda M})^2(1-\mathrm{e}^{-2\lambda E})}{8\lambda^3M^2E}. \end{aligned} \quad (5)$$

Now, we obtain the deadline meet ratio of the LST scheduling algorithm when the task A and task B conflict.

$$\begin{aligned}\mathrm{DM}_{\mathrm{lst}} &= \mathrm{Mean}(\mathrm{DM}_A(2\lambda, m)) \\ &\quad + \left\{\frac{2(1-\mathrm{e}^{-\lambda M})^2(1-\mathrm{e}^{-\lambda E})}{\lambda^3M^2E} - \frac{(1-\mathrm{e}^{-2\lambda M})^2(1-\mathrm{e}^{-2\lambda E})}{8\lambda^3M^2E}\right\}\Big/2 \\ &= \frac{1}{2\lambda M}(1-\mathrm{e}^{-2\lambda M}) \\ &\quad + \left\{\frac{2(1-\mathrm{e}^{-\lambda M})^2(1-\mathrm{e}^{-\lambda E})}{\lambda^3M^2E} - \frac{(1-\mathrm{e}^{-2\lambda M})^2(1-\mathrm{e}^{-2\lambda E})}{8\lambda^3M^2E}\right\}\Big/2. \end{aligned} \quad (6)$$

2.3. ELST (effective least slack time first)

Preemptive LST is an optimal algorithm in real-time scheduling algorithm. However, in CPS, we need to consider physical environments to improve the deadline meet ratio. As an example, we have to consider the moving time of computing (servicing) node to the location of task serviced. Let $l_{\mathrm{eff},i}$ be an effective slack time of task i (slack time including moving time), then $l_{\mathrm{eff},i}$ is computed as follows:

$$l_{\mathrm{eff},i} = l_i - m_i. \quad (7)$$

Now, we compute the $l_{\mathrm{eff},i}$. As the distribution of l_i is $\lambda\mathrm{e}^{-\lambda t}$, $l_{\mathrm{eff},i}$ (when $l_{\mathrm{eff},i} > 0$) distribution is computed as follows:

$$\frac{1}{M}\int_0^M \lambda\mathrm{e}^{-\lambda(t+m)}dm = \frac{\mathrm{e}^{-\lambda t}}{M}(1-\mathrm{e}^{-\lambda M}). \quad (8)$$

$l_{\mathrm{eff},i}$ (when$-M < l_{\mathrm{eff},i} < 0$) the distribution is computed as follows:

$$\frac{1}{M}\int_{-t}^M \lambda\mathrm{e}^{-\lambda(t+m)}dm = \frac{1}{M}(1-\mathrm{e}^{-\lambda(t+M)}). \quad (9)$$

An average deadline meet ratio of task A (without conflict) is the probability of $l_{\mathrm{eff},i} > 0$.

$$\begin{aligned}\mathrm{Mean}(\mathrm{DM}_A(\lambda, m)) &= \int_0^\infty \frac{\mathrm{e}^{-\lambda t}}{M}(1-\mathrm{e}^{-\lambda M})dt \\ &= \frac{1}{\lambda M}(1-\mathrm{e}^{-\lambda M}). \end{aligned} \quad (10)$$

The deadline meet ratio of task B following task A is:

$$\begin{aligned}\mathrm{DM}_B(\lambda, m_A, e_A, m_B) &= \int_{m_A+e_A}^\infty \frac{\mathrm{e}^{-\lambda t}}{M}(1-\mathrm{e}^{-\lambda M})dt \\ &= \frac{(1-\mathrm{e}^{-\lambda M})}{\lambda M}\mathrm{e}^{-\lambda(m_A+e_A)}. \end{aligned} \quad (11)$$

As we assume m_A and e_A are evenly distributed on $[0, M]$ and $[0, E]$, respectively, mean (DM_B (λ, m_A, e_A, m_B)) is computed as:

$$\begin{aligned}\mathrm{Mean}(\mathrm{DM}_B(\lambda, m_A, e_A, m_B)) &= \frac{1}{ME}\int_0^E\int_0^M \frac{(1-\mathrm{e}^{-\lambda M})}{\lambda M} \\ &\quad \times \mathrm{e}^{-\lambda(m_A+e_A)}dm_A de_A \\ &= \frac{(1-\mathrm{e}^{-\lambda M})^2(1-\mathrm{e}^{-\lambda E})}{\lambda^3M^2E}. \end{aligned} \quad (12)$$

We can find that mean (DM_A (λ, m)) and mean DM_B (λ, m_A, e_A, m_B) are same as those obtained in Section 2.1. As parameters using in two analyses are the same but $l_{\mathrm{eff},i} = l_{i,} - m_i$, two deadline meet ratios computed in 2.1 and 2.3 must be the same. (One uses $l_i > m_i$ while the other $l_{\mathrm{eff},i} = (l_{i,} - m_i) > 0$, which is basically same, to compute the deadline meet ratio.)

2.4. O-ELST (optimal effective least slack time first)

As the ELST algorithm cannot improve the real-time performance, we consider the optimal algorithm which changes schedule for two tasks (task A and task B) when the changed schedule can improve the real-time performance. Let p be the probability of meeting the deadline of firstly scheduled task (task A).

$$\begin{aligned}P &= \mathrm{Mean}(\mathrm{DM}_A(\lambda, m)) = \int_0^\infty \frac{\mathrm{e}^{-\lambda t}}{M}(1-\mathrm{e}^{-\lambda M})dt \\ &= \frac{1}{\lambda M}(1-\mathrm{e}^{-\lambda M}). \end{aligned} \quad (13)$$

Table 2
O-ELST scheduling algorithm when task A and task B conflict.

Deadline meet/miss on schedule $A \rightarrow B$ (A followed by B)	Probability	O-ELST schedule	Probability of O-ELST choosing this schedule	No. of task meeting deadline
meet A, meet B	$p * q$	$A \rightarrow B$	$p * q$	2
meet A, miss B	$p(1-q)$	Change schedule $B \rightarrow A$ if meet both A and B	$p(1-q) * (p-q)/(1-q) * q/p$	2
		$A \rightarrow B$if $B \rightarrow A$ is not better	$p(1-q) * \{1-(p-q)/(1-q) * q/p\}$	1
miss A, meet B	$(1-p)q$	$A \rightarrow B$ ($B \rightarrow A$)	$(1-p)q$	1
miss A, miss B	$(1-p) * (1-q)$	Schedule $B \rightarrow A$ if meet B	$(1-p)(1-q) * (p-q)/(1-q)$	1
		$A \rightarrow B$(if $B \rightarrow A$ is not better)	$(1-p)(1-q) * \{1-(p-q)/(1-q)\}$	0

Let q be the probability of meeting the deadline of the secondly scheduled task (task B).

$$q = \mathrm{Mean}(\mathrm{DM}_B(\lambda, m_A, e_A, m_B)) = \frac{1}{ME}\int_0^E \int_0^M \frac{(1-e^{-\lambda M})}{\lambda M} e^{-\lambda(m_A+e_A)} dm_A de_A = \frac{(1-e^{-\lambda M})^2(1-e^{-\lambda E})}{\lambda^3 M^2 E}. \quad (14)$$

We use somewhat different approach from LST and ELST scheduling to compute the deadline meet ratio for O-ELST scheduling. O-ELST scheduling considers the moving time as well as the slack time to improve the deadline meet ratio. When task A and task B conflict, O-ELST schedules tasks (A followed by B or B followed by A), which maximizes a deadline meet ratio. On a schedule of A followed by B, there are four cases:

- Both A and B meet the deadline (probability of pq): in this case, O-ELST does not change the schedule (choose the schedule of A followed by B).
- A only meets the deadline (probability of $p(1-q)$): in this case, O-ELST changes schedule (B followed by A) if both A and B meet the deadline. Probability of meeting deadline for both A and B by changing schedule is $(p-q)/(1-q) * q/p$. ((probability of B meeting the deadline at scheduling of B followed by A on the condition of missing the deadline at scheduling of A followed by B) $*$ (probability of A meeting the deadline also even at scheduling of B followed by A on the condition of meeting the deadline at scheduling of A followed by B)).
- B only meets the deadline (probability of $(1-p)q$): in this case, A cannot meet deadline at any scheduling.
- Neither A nor B meets the deadline (probability of $(1-p)$ $(1-q)$): in this case, O-ELST changes schedule (B followed by A) if B meets the deadline. Probability of meeting the deadline for B by changing schedule is $(p-q)/(1-q)$ (probability of B meeting the deadline at scheduling of B followed by A on the condition of missing the deadline at scheduling of A followed by B). In this case, A cannot meet deadline at any scheduling.

The other schedule, B followed by A, has also four cases. O-ELST chooses the schedule which maximizes the deadline meet ratio by considering the moving time as well as the slack time.

From Table 2, we can obtain the deadline meet ratio of O-ELST scheduling algorithm when task A and task B conflict. We can compute the expected number of tasks meeting the deadline by summation of products of columns "probability of O-ELST choosing this schedule" and "number of task meeting deadline". After that, the deadline meet ratio is the half of the expected number of tasks meeting the deadline as there are two

$$\begin{aligned}\mathrm{DM}_{\text{o-elst}} &= [2pq + 2p(1-q) * (p-q)/(1-q) * q/p \\ &\quad + p(1-q) * \{1-(p-q)/(1-q) * q/p\} \\ &\quad + (1-p)q + (1-p)(1-q) * (p-q)/(1-q)]/2 \\ &= (2p - p^2 + 2pq - q^2)/2,\end{aligned}$$

where $p = \mathrm{Mean}(\mathrm{DM}_A(\lambda, m)) = \frac{1}{\lambda M}(1-e^{-\lambda M})$ and

$$q = \mathrm{Mean}(\mathrm{DM}_B(\lambda, m_A, e_A, m_B)) = \frac{(1-e^{-\lambda M})^2(1-\varepsilon e^{-\lambda E})}{\lambda^3 M^2 E}. \quad (15)$$

2.5. Heuristic ELST(H-ELST) algorithm

In the Section 2.3, we analyzed the performance of O-ELST only with two serviced nodes. However, this method is not practical because time complexity becomes huge as the number of nodes increases. Now, we briefly explain H-ELST (Heuristic-ELST) algorithm to reduce the time complexity while maintaining the deadline meet ratio. Real-Time H-ELST scheduling algorithm is as follows:

- modify the LST scheduling algorithm by weighting on the slack time and physical migration time;
- give priority to serviced nodes with not only small slack time but also small moving time: as the weighted sum of slack time and moving time decreases, the priority increases;
- give "α" ($0 < \alpha < 1$) weight to the slack time and "$1-\alpha$" weight to the migration time. (Performance will depend on the value of the weight parameter α;)
- focus on physical factors (migration delay time, etc.) as well as cyber factors (period, execution time, release time, deadline, etc.).

3. Performance comparisons for real-time CPS

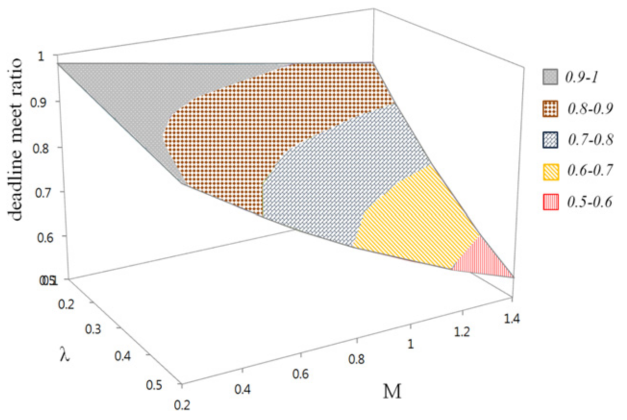
3.1. Real-time performance analysis by mathematical analysis

We measure the performance by varying parameters, λ and M. (We assume that $M = E$ for easy analysis and comparison.) We compare the performance among FIFO, LST, and O-ELST. Fig. 2 shows deadline meet ratios for FIFO, LST, and O-ELST scheduling algorithms. Fig. 3 shows relative views of Fig. 2 (relative deadline miss ratios of O-ELST to FIFO and O-ELST to LST). O-ELST algorithm can reduce deadline meet ratios up to 49% and 22% comparing to FIFO and LST algorithms, respectively.

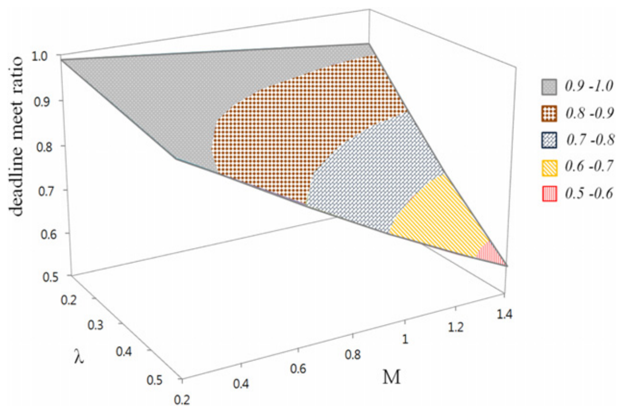
3.2. Real-time performance analysis by simulation

We measure the performance of the real-time scheduling algorithms by simulation to verify the performance of the proposed real-time scheduling for CPS in the general cases of many tasks conflicting. We use the parameters for real-time scheduling algorithm in the simulation as follows:

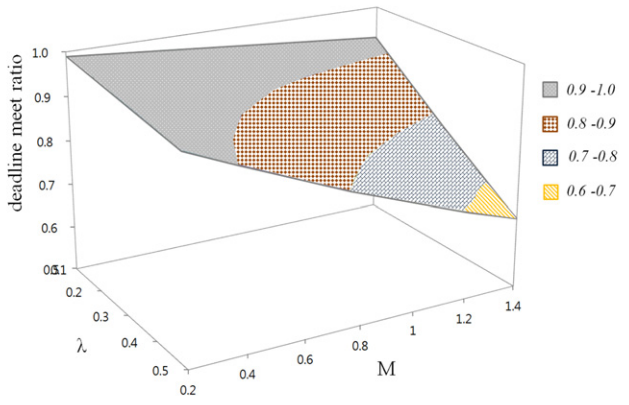
- n: number of nodes (tasks) requiring real-time service, $2 < n < 8$
- n_s: number of nodes performing real-time service, $n_s = 1$
- α: weight of execution time (weight of migration time is $1-\alpha, 0 < \alpha < 1$)



(a) Deadline meet ratio (FIFO).

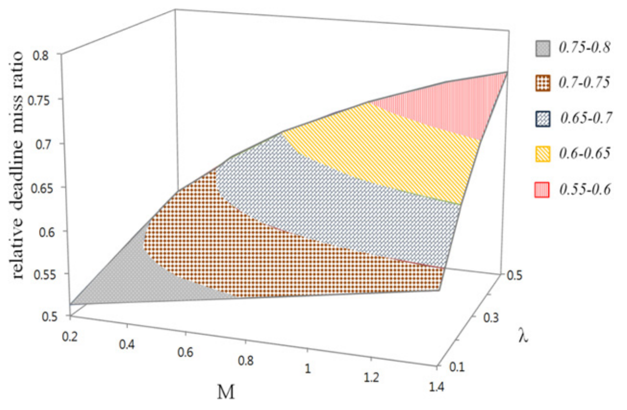


(b) Deadline meet ratio (LST).

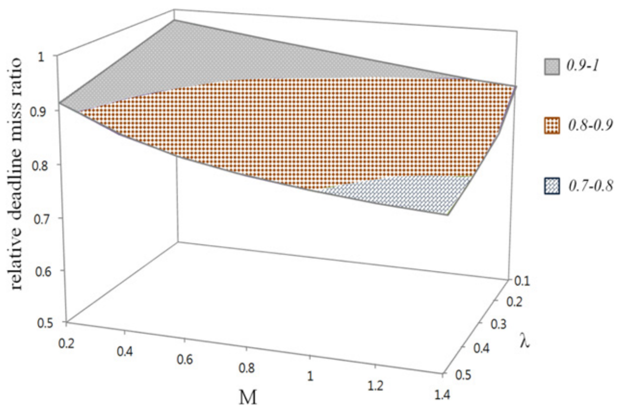


(c) Deadline meet ratio (O-ELST).

Fig. 2. Deadline meet ratios for FIFO, LST, and O-ELST scheduling algorithms.



(a) Relative deadline miss ratio (O-ELST to FIFO).



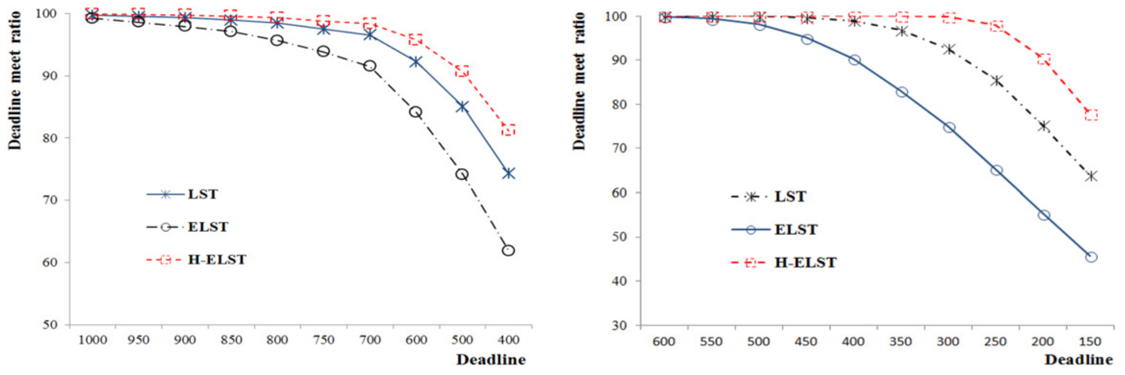
(b) Relative deadline miss ratio (O-ELST to LST).

Fig. 3. Relative deadline miss ratios of O-ELST to FIFO and O-ELST to LST.

- e_i: execution time at node i with exponential distribution of average $1/\lambda$ ($\lambda = 0.018$) (Fig. 4(a)), $\lambda = 0.2$ (Fig. 4(b))
- d_i: deadline at the node i with variable range
- m_{ci}: moving distance between computing node c and serviced node i ($m_{ci} = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}$, position of node i (x_i, y_i) and position of computing node c (x_c, y_c) are evenly distributed in a square of 100 by 100)
- r_i: release time, $r_i = 0$
- number of simulations: 10 000 times.

We compare the performance of our algorithm in terms of a deadline meet ratio with LST, ELST, and H-ELST in the general case of many tasks conflicting. The results of performance evaluation results are shown in Fig. 4.

Although the results differ depending on "α", H-ELST shows excellent performance for deadline meet ratio. (Values between 0.1–0.9 are compared and "α" of the best performance is selected in the simulation. The value 0.6 is applied in this simulation. Future work is required to find a proper weight of "α").



(a) When cyber factor is quite big in comparison to physical factor. (b) When cyber factor is quite small in comparison to physical factor.

Fig. 4. Deadline meet ratios for LST, ELST and H-ELST scheduling algorithms.

ELST algorithm and LST algorithm show difference in performance depending on the ratio of the cyber factor (e_i) and the physical factor (m_i). As shown in Fig. 4, depending on the ratio of cyber factor (e_i) and the physical factor (m_i), the conventional LST algorithm shows better performance than the ELST algorithm at certain conditions. Although not included in the paper, the EEDF(Effective Earliest Deadline First) algorithm shows better performance than the EDF algorithm in the simulation when the cyber factor (e_i) is much larger than the physical factor (m_i).

Summarizing the simulation results, ELST and EEDF scheduling algorithms show good performance for deadline meet ratio when the cyber factor (e_i) is much larger than the physical factor (m_i). On the other hand, when the physical factor (e_i) is much larger than the cyber factor (m_i), the conventional LST and EDF algorithms show better performance than the ELST and EEDF algorithms, respectively. However, the H-ELST algorithm applied with a proper "α" weight shows excellent performance in many conditions. As a result, H-ELST scheduling algorithms can reduce deadline meet ratios by up to 20% compared to LST.

4. Conclusions and future work

As conventional real-time scheduling algorithm considers system resources in cyber space such as CPU, network bandwidth, and memory, it is not proper to apply in CPS. We propose a real-time scheduling algorithm for cyber physical systems society, where physical factors (e.g., location, migration delay time, etc.) affect the real-time performance. To demonstrate the real-time scheduling algorithm for CPS, we assume a simple CPS environment in which the computing node moves around physically distributed tasks to perform real-time services. Performance measurement by mathematical analysis and simulations shows that O-ELST (Optimal Effective Least Slack Time First) algorithm reduces a deadline miss ratio up to 49% and 22% comparing to FIFO (First In First Out) and LST (Least Slack Time First), respectively. Simulation results also show that the real-time scheduling algorithm for CPS (H-ELST: Heuristic Effective Least Slack Time First) can improve the performance by including physical factor (moving time). As a future work, we plan to perform extensive simulations to verify the performance of H-ELST in more realistic environment. In addition, we will study various algorithms for CPS considering social factors.

Acknowledgments

This research is supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2012R1A1A2A10041537).

References

[1] B. Gerkey, S. Chitta, M. Beetz, L.E. Kavraki, Real-time guidance under MRI, IEEE Engineering in Medicine and Biology Magazine 29 (2) (2010) 78–86.

[2] C. Domínguez, H. Hassan, A. Crespo, Real-time embedded architecture for improving pervasive robot services, International Journal of Software Engineering and its Application 2 (1) (2008) 79–90.

[3] S.-H. Park, J.-H. Kim, C.-H. Han, K.-S. Kim, Effective one-to-one correspondence method of $O(N^2\mathrm{Log}(N))$ complexity between distributed units, International Journal of Advanced Robotic Systems 9 (2012) 1–7.

[4] S.-H. Park, J.-H. Kim, C.-H. Han, The extended-military multimedia systems based on real-time scheduling scheme, Journal of the Institute of Electronics Engineers of Korea 48-CI (1) (2011) 26–32.

[5] C.-H. Han, Y.-H. Min, S.-H. Park, J.-H. Kim, Minimum-cost path finding algorithm in real-time for computer generated force, Journal of the Institute of Electronics Engineers of Korea 48-CI (1) (2011) 17–25.

[6] S.-H. Park, J.-H. Kim, Real-time NCW systems using distributed processing, in: Kisse, Korea Computer Congress, Vol. 36, 2009, pp. 245–249.

[7] W.H. Yuan, K. Nahrstedt, Energy-efficient soft real-time CPU scheduling for mobile multimedia systems, in: SOSP'03 Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, Vol. 37, No. 5, 2003, pp. 149–163.

[8] A. Abdelli, A much compact abstraction of the state space of real time preemptive systems, International Journal of Advanced Science and Technology 27 (2011) 45–58.

[9] J. Wang, S. Han, K.-Y. Lam, K. Mok, Maintaining data temporal consistency in distributed real-time systems, Real-Time Systems 48 (4) (2012) 387–429.

[10] Q.Y. Dai, R.Y. Zhong, M.L. Wang, X.D. Liu, Q. Liu, RFID-enable real-time multi-experiment training center management system, International Journal of Advanced Science and Technology 7 (2009) 27–48.

[11] C.L. Liu, J. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, Journal of the ACM 20 (1973) 46–61.

[12] A. Burns, Scheduling hard real-time systems: a review, Software Engineering Journal 6 (1991) 116–128.

[13] Z.R.M. Azmi, K.A. Bakar, M.S. Shamsir, W.N.W. Manan, A.H. Abdullah, Performance comparison of priority rule scheduling algorithms using different inter arrival time jobs in grid environment, International Journal of Advanced Science and Technology 4 (3) (2011) 61–70.

[14] J.W.S.W. Liu, Real-Time Systems, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[15] E.A. Lee, CPS foundations, in: DAC'10 Proceedings of the 47th Design Automation Conference, June 13–18, Anaheim, California. USA, 2010, pp. 737–742.

[16] R.R. Rajkumar, I.S. Lee, L. Sha, J. Stankovic, Cyber-physical systems: the next computing revolution, in: DAC'10 Proceedings of the 47th Design Automation Conference, June 13–18, Anaheim, California. USA, 2010, pp. 731–736.

[17] J.C. Eidson, E.A. Lee, S. Matic, S.A. Seshia, J. Zou, Distributed real-time software for cyber-physical systems, Proceedings of the IEEE 100 (2011) 45–59.

[18] E.A. Lee, Computing needs time, Communications of the ACM 52 (5) (2009) 70–79.

[19] Hai Zhuge, Cyber physical society, in: Sixth International Conference on Semantics, Knowledge and Grids, 2010, pp. 1–8.

[20] Hai Zhuge, Semantic linking through spaces for cyber-physical-socio intelligence: a methodology, Artificial Intelligence 175 (2011) 988–1019.

[21] Hai Zhuge, Yunpeng Xing, Probabilistic resource space model for managing resources in cyber-physical society, IEEE Transactions on Services Computing 5 (3) (2012) 404–421.

[22] A. Banerjee, Ensuring safety, security, and sustainability of mission-critical cyber-physical systems, Proceedings of the IEEE 100 (1) (2012) 283–299.

[23] J.-H. Kim, S.-H. Park, Geoffery Fox, Real-time scheduling in cyber-physical systems, in: International Conference, CCA, 2012, p. 69.

[24] Tiziana Ferrari, Luciano Gaido, Resources and services of the EGEE production infrastructure, Journal of Grid Computing 9 (2) (2011) 119–133.
[25] Antun Balaž, et al., Development of grid e-Infrastructure in South-Eastern Europe, Journal of Grid Computing 9 (2) (2011) 135–154.
[26] Francisco Brasileiro, et al., Using a simple prioritisation mechanism to effectively interoperate service and opportunistic grids in the EELA-2 e-Infrastructure, Journal of Grid Computing 9 (2) (2011) 241–257.
[27] Bryan Che, Red hat enterprise MRG: messaging, realtime, and grid, www.redhat.com.
[28] Sangho Yi, et al. Towards real-time, volunteer distributed computing, in: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid, 2011, pp. 154–163.

Sanghyuk Park received his B.S. degree in Ocean Engineering from Korea Maritime University, Busan, South Korea, in 2000 and his Ph.D. degree in NCW (Network Centric Warfare), from Ajou University, Suwon, Korea, in 2013. He is a Marine Corps officer of the Republic of Korea. His main research interests include NCW (Network Centric Warfare), Real-Time System, Distributed System, Mobile Computing and Cyber Physical Systems.

Jai-Hoon Kim received his B.S. degree in Control and Instrumentation Engineering from the Seoul National University, Seoul, South Korea, in 1984, M.S. degree in Computer Science from the Indiana University, USA, in 1993, and his Ph.D. degree in Computer Science from the Texas A&M University, USA, in 1997. He is currently a professor at the Information and Computer Engineering department, Ajou University, South Korea. His research interests include Distributed Systems, Real-Time systems, and Mobile Computing.

Geoffrey C. Fox received a Ph.D. in Theoretical Physics from Cambridge University and is now a professor of Computer Science, Informatics, and Physics at Indiana University. He is the director of the Community Grids Laboratory of the Pervasive Technology Laboratories at Indiana University. He previously held positions at Caltech, Syracuse University and Florida State University. He has published over 600 papers in physics and computer science and been a major author on four books. Fox currently works in applying computer science to Defense, Earthquake and Ice-sheet Science and Chemical Informatics. He is involved in several projects to enhance the capabilities of Minority Serving Institutions. His main research interests include Practical Computer Science with applications.