**Effective Real-Time Scheduling Algorithm for Cyber Physical Systems Society**

Sanghyuk Park1 , Jai-Hoon Kim2\*, Geoffrey Fox3

1Department of Network Centric Warfare Engineering, Ajou University, S. Korea

2Graduate School of Information and Communication, Ajou University, S. Korea

3Pervasive Technology Institute, Indiana University, Bloomington, Indiana, U.S.A.

***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 cyber space (CPU, network, storage systems, etc.) and physical space (location, migration, etc.) but also socio space and mental space for 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 real-time performance in terms of deadline meet ratio by mathematics analysis and simulations. The results show that our algorithms reduces a deadline miss ratio approximately up to 50% and 20% comparing to conventional real-time scheduling algorithm, FIFO (First In First Out) and LST (Least Slack Time First), respectively.

***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)***

**1 Introduction**

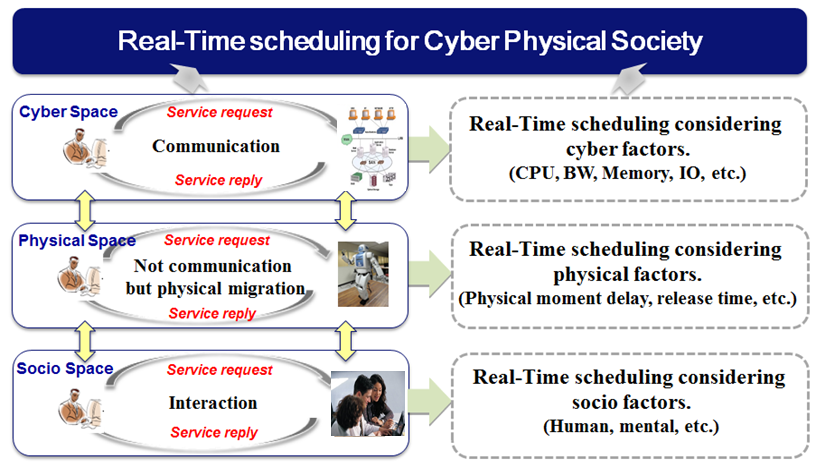
Timing issues are critical in real-time systems such as robot control [1, 2], NCO (Network Centric Operations) systems [3, 4, 5, 6], flight control, on-line multimedia systems [7], and real-time stock trading system, etc. [8,9,10]. Many real-time scheduling algorithms such as RM (rate monotonic) [11,12], EDF (earliest deadline first) [12, 13, 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, 16, 17, 18] and cyber physical society [21, 22, 23]) 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 new research issues in the real-time systems [19, 20].

In other aspects, as real-time applications become complex and relevant tasks and resources are widely distributed, we have to study real-time scheduling in distributed computing infrastructures and Grids. For examples, in Grids infrastructures (e.g., EGI (European Grid Infrastructure) [28], SEE-GRID (South Eastern European Grid-enabled einfrastructure) [29], and EELA (E-science grid facility for Europe and Latin America) [30]) many tasks concurrently request various types of distributed resources. Middleware has to coordinate resource allocation to provide services and guarantee a SLA. In these distributed environments, real-time scheduling must considers transfer delays as task and data migrations among nodes having computing resources are common. Red Hat Enterprise MRG (Messaging, Realtime, and Grid) Realtime [31] 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. Reference [32] design and implement the real-time volunteer computing platform called RT-BOINC to schedule real-time task and execute on the volunteer resources.

\* Corresponding Author.

*E-mail addresses* : [jaikim@ajou.ac.kr](mailto:jaikim@ajou.ac.kr) (Jai-Hoon Kim)

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 [21, 22, 23]. Fig. 1 shows the real-time scheduling model for cyber physical systems society [21, 22, 23]. Effective release time and deadline of real-time tasks may be different depending on 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 effective release time and deadline.



**Fig. 1.** Real-time Scheduling for CPS

In this paper, we propose new real-time scheduling algorithms for CPS, where 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 performance (e.g., deadline meet ratio). In this case, 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 conventional real-time scheduling and CPS real-time scheduling. As in many kinds of CPS, where servicing nodes must move to 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 references [21, 22, 23] such as socio space, mental space, etc.

**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. | Migration is required  for CPU and Job. |
| \* We consider CPU is serving node and Job is serviced node | |
| 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) |

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 real-time performance (deadline meet ratio) between conventional real-time scheduling and proposed real-time scheduling for CPS. We assume parameters for real-time systems as follows:

• *li* : slack (laxity) time of task i (exponential distribution of average 1/λ)

• *ei* : execution time of task i (evenly distributed on [0, E])

• *mi* : 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 slack time *lA* being greater than moving time mA (servicing node moving to serviced node (task A) within slack time *lA*). As distribution of *lA* is λ, deadline meet ratio of a task A (DMA (λ, m)) is computed as follow:

(1)

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

(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 subsection 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 task A arrived just before the other task B. A deadline meet ratio of task A is mean (DMA (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 (*lB*) being greater than *mA* + e*A* + *mB*. Thus, deadline meet ratio of task B following task A (Mean (DMB (λ, *mA*, e*A*, *mB*))) is computed as follow:

(3)

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

(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 exponential distribution (2λ) of average 1/(2λ) while the slack of task B is exponential distribution of 2λ–2λ = 2λ (1 - λ). (We can obtain it by using Markov model.) A deadline meet ratio of task A is mean (DMA (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 (*lB*) being greater than *mA* + e*A* + *mB*. Thus, an average deadline meet ratio of task B following task A is computed as follow:

(5)

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

(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 deadline meet ratio. As an example, we have to consider moving time of computing (servicing) node to the location of task serviced. Let’s denote *leff, i* be an effective slack time of task i (slack time including moving time), then *leff, i* is computed as following:

*leff, i* = *li - mi.*

(7)

Now, we compute the *leff, i*. As distribution of *l i* is λ, *leff, i* (when *leff, i* > 0) distribution is computed as follow:

(8)

*leff, i* (when –M < *leff, i* <0) distribution is computed as follow:

(9)

An average deadline meet ratio of task A (without conflict) is the probability of *leff, i* > 0.

(10)

Deadline meet ratio of task B following task A is:

(11)

As we assume *mA* and *eA* are evenly distributed on [0, M] and [0, E], respectively, mean (D*MB* (*λ*, *mA*, *eA*, *mB*)) is computed as:

(12)

We can find that mean (DMA (λ, m)) and mean (D*MB* (*λ*, *mA*, *eA*, *mB*) are same as those obtained in subsection 2.1. As parameters using in two analyses are the same but *leff, i* = *li, - mi*, two deadline meet ratios computed in 2.1 and 2.3 must be the same. (One uses *li* > *mi* while the other *leff, i = (li,-mi) > 0*, which is basically same, to compute deadline meet ratio.)

**2.3 O-ELST (Optimal Effective Least Slack Time First)**

As the ELST algorithm cannot improve 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 probability of meeting deadline of firstly scheduled task (task A).

(13)

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

(14)

We use somewhat different approach from LST and ELST scheduling to compute deadline meet ratio for O-ELST scheduling. O-ELST scheduling considers moving time as well as slack time to improve 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 schedule (choose 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 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 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 maximize deadline meet ratio by considering moving time as well as slack time.

**Table 2**

O-ELST scheduling algorithm when task A and task B conflict

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Deadline meet/miss  on schedule A🡺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🡺B | p\*q | 2 |
| meet A, miss B | p(1-q) | change schedule B🡺A  if meet both A and B | p(1-q)\*(p-q)/(1-q)\*q/p | 2 |
| A🡺B  If B🡺A is not better | p(1-q)\*{1-(p-q)/(1-q)\*q/p} | 1 |
| miss A, meet B | (1-p)q | A🡺B (B🡺A) | (1-p)q | 1 |
| miss A, miss B | (1-p)\*(1-q) | schedule B🡺A if meet B | (1-p)(1-q)\*(p-q)/(1-q) | 1 |
| A🡺B  (if B🡺A is not better) | (1-p)(1-q)\*{1-(p-q)/(1-q)} | 0 |

From the above table, we can obtain the deadline meet ratio of O-ELST scheduling algorithm when task A and task B conflict. We can compute 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, deadline meet ratio is the half of expected number of tasks meeting the deadline as there are two tasks.

*DMo-elst = [2pq + 2p (1-q)\*(p-q)/(1-q)\*q/p + p (1-q)\*{1-(p-q)/(1-q)\*q/p}*

*+ (1-p) q + (1-p) (1-q)\*(p-q)/(1-q)]/2= (*

(15)

**2.4 Heuristic ELST(H-ELST) Algorithm**

In the subsection 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 node increases. Now, we briefly explain H-ELST (Heuristic-ELST) algorithm to reduce time complexity while maintaining deadline meet ratio. Real-Time H-ELST scheduling algorithm is as follows;

• Modify the LST scheduling algorithm by weighting on slack time and physical migration time.

• Give priority for 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.

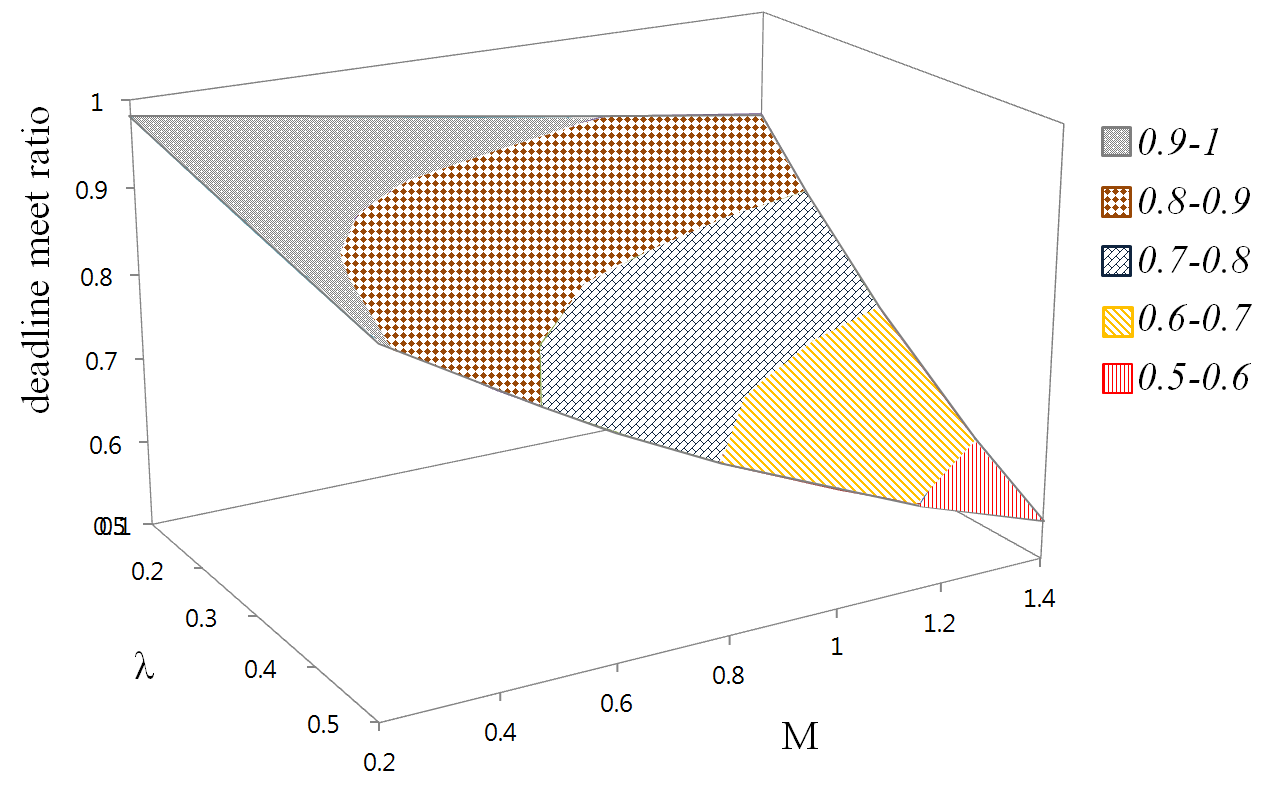
• Give “α” (0 < α < 1) weight to slack time and “1 – α” weight to migration time. (Performance will depend on the value of the weight parameter α.)

• Focus on physical factors (migration delay time, etc.) as well as cyber factors (period, execution time, release time, deadline, etc.).

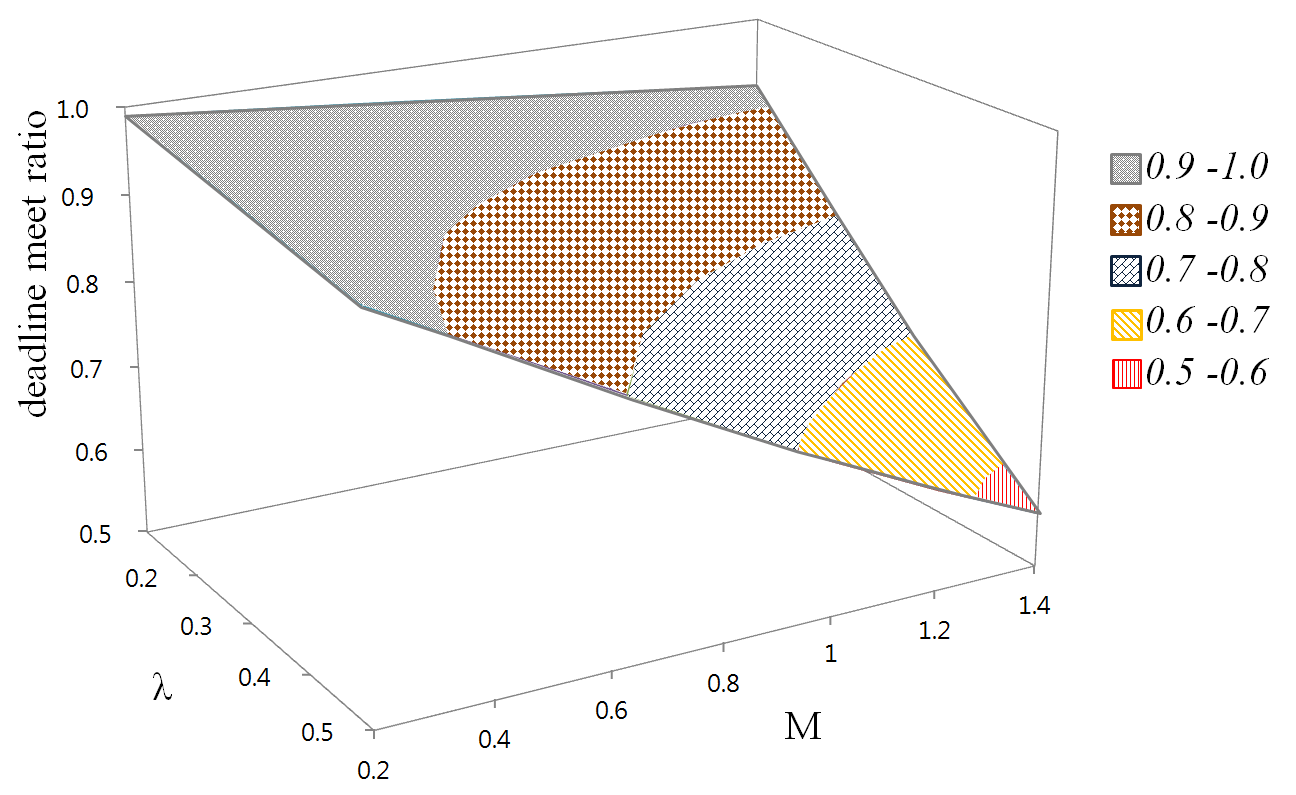
**3 Performance Comparisons for Real-Time CPS**

**3.1 Real-Time performance analysis by mathematical analysis**

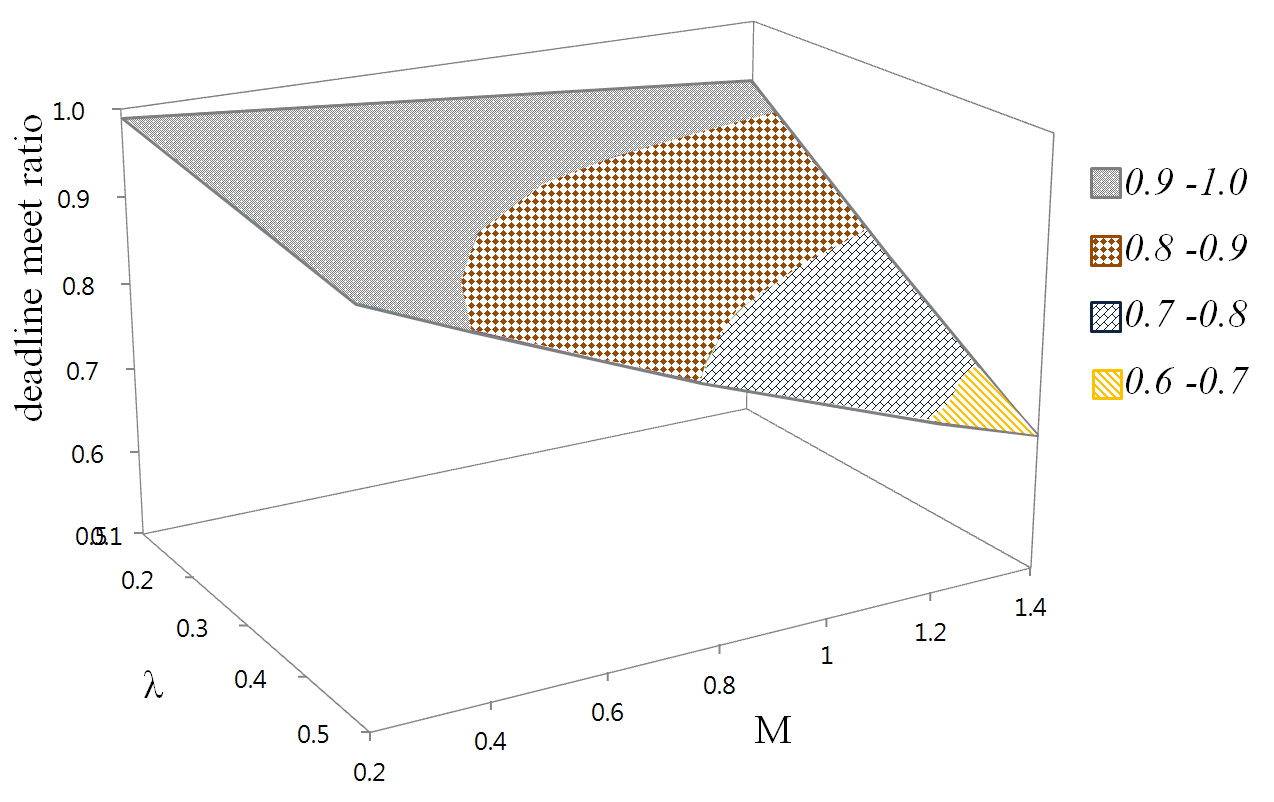
We measure performance by varying parameters, λ and M. (We assume that M=E for easy analysis and comparison.) We compare performance among FIFO, LST, and O-ELST. Fig. 2 shows deadline meet ratios for FIFO, LST, and O-ELST scheduling algorithms. Fig. 3 shows relative views of Fig. 2 (relative deadline 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.



(a) Deadline meet ratio (FIFO)

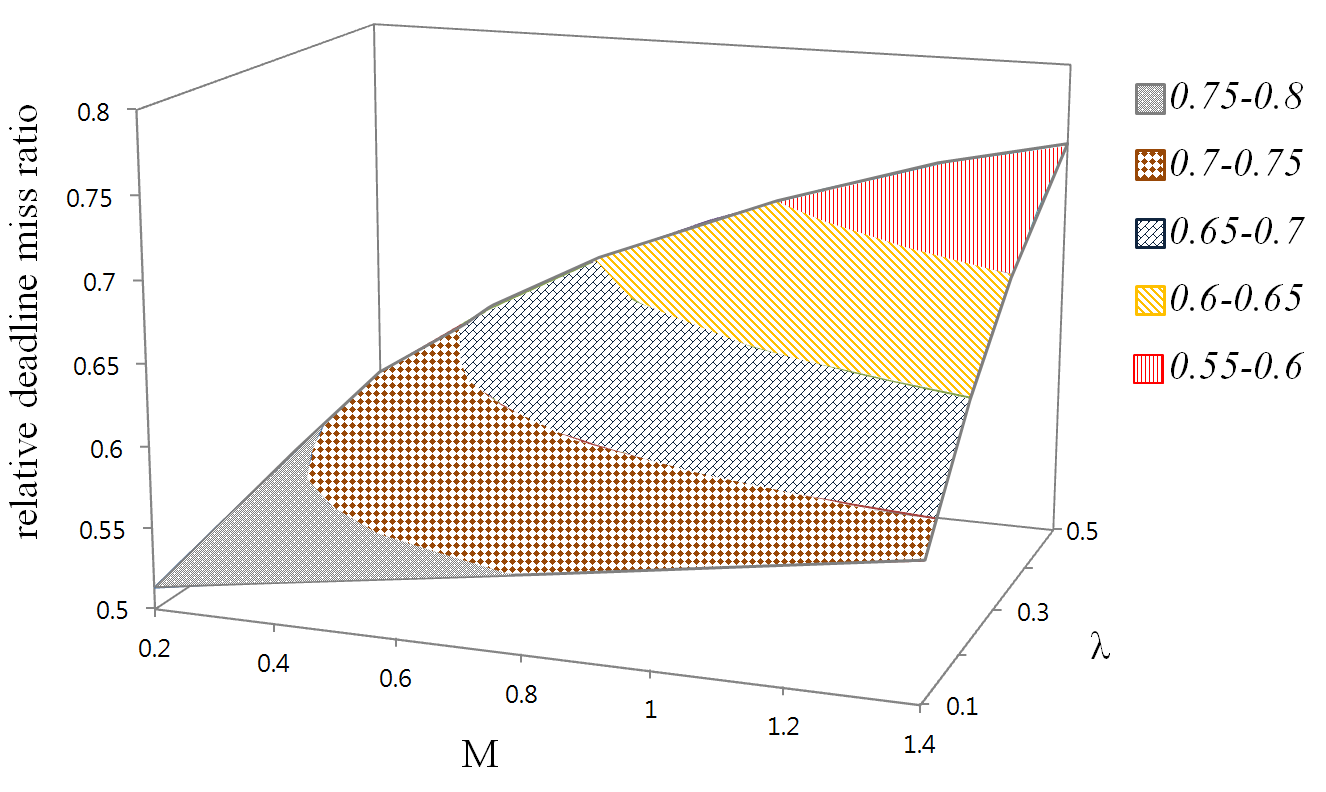


(b) Deadline meet ratio (LST)

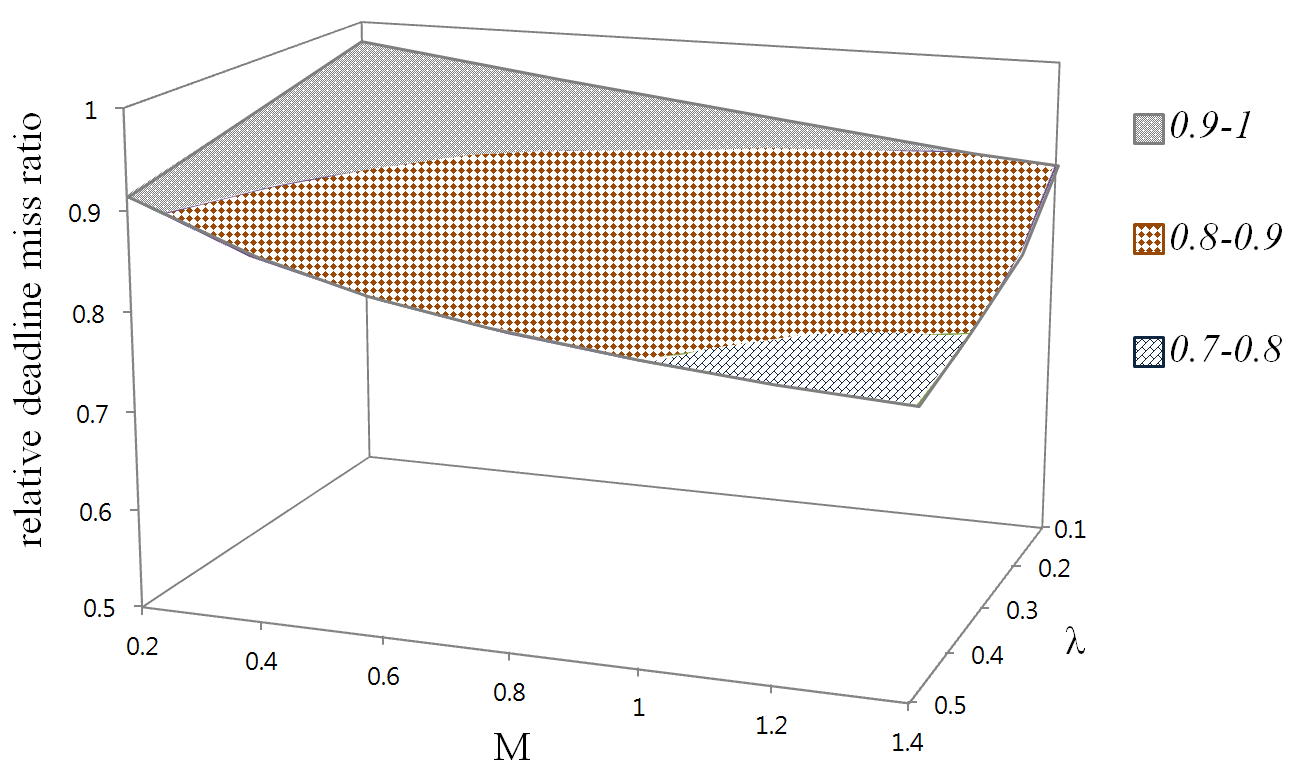


(c) Deadline meet ratio (O-ELST)

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



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



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

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

**3.2 Real-Time performance analysis by simulation**

We measure the performance of real-time scheduling algorithms by simulation to verify the performance of 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*

*• ns: number of nodes performing real-time service, ns = 1*

*• α: weight of execution time (weight of migration time is 1- α, 0 < α <1)*

*• ei: execution time at node i with exponential distribution of average 1/λ*

*(λ=0.018(Fig. 4 (a), λ=0.2(Fig. 4(b))*

*• di: deadline at the node i with variable range*

*• mci : moving distance between computing node c and serviced node i*

*(mci = , position of node i (xi ,yi) and position of computing node c (xc ,yc) are evenly distributed in a square of 100 by 100)*

*• ri: release time, ri = 0*

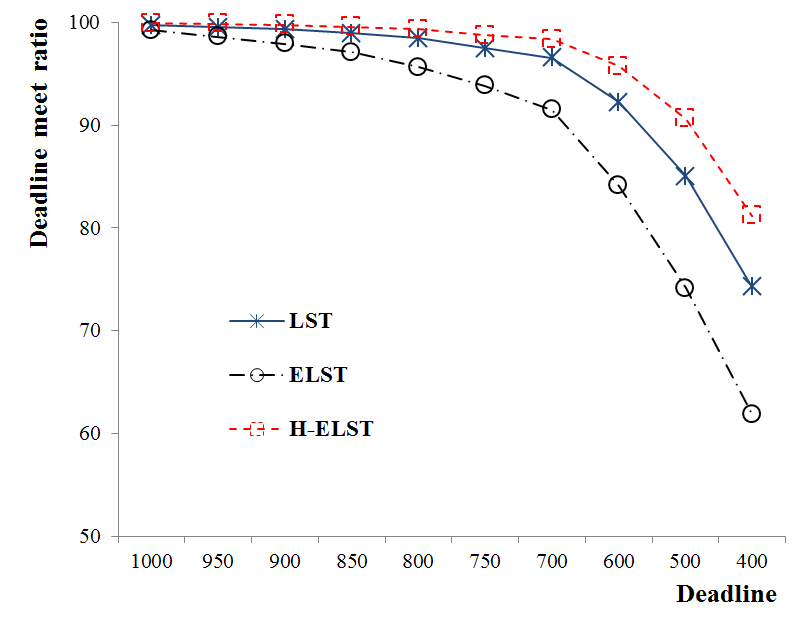
*• number of simulations: 10,000 times*

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.

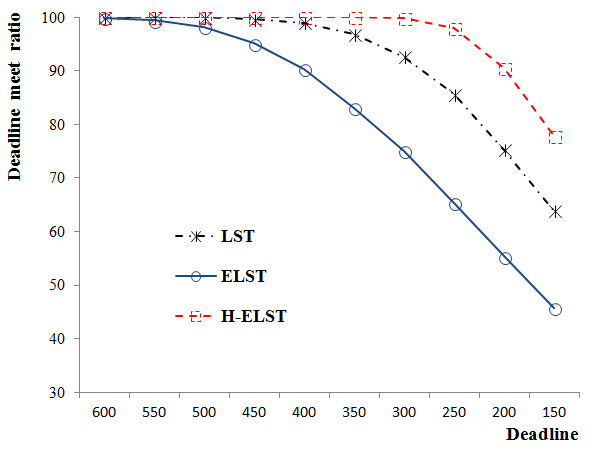
Although the results differ depending on “α”, H-ELST shows excellent performance for deadline meet ratio. (Values between 0.1~0.9 are compared and “α” of the best performance is selected in the simulation. The value 0.6 is applied in this simulation. Future work is required to find a proper weight of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

ELST algorithm and LST algorithm show different in performance depending on the ratio of the cyber factor (ei) and the physical factor (mi). As shown in Fig. 4, depending on the ratio of cyber factor (ei) and the physical factor (mi), 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 (ei) is much larger than the physical factor (mi).

Summarizing the simulation results, ELST and EEDF scheduling algorithms show good performance for deadline meet ratio when the cyber factor (ei) is much larger than the physical factor (mi). On the other hand, when the physical factor (ei) is much larger than the cyber factor (mi), 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.



1. When cyber factor is quite big in comparison to physical factor



1. When cyber factor is quite small in comparison to physical factor

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

**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 does 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 on real-time performance. To demonstrate real-time scheduling algorithm for CPS, we assume a simple CPS environment in which computing node moves around physically distributed tasks to perform real-time services. Performance measurement by mathematics 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 real-time scheduling algorithm for CPS (H-ELST: Heuristic Effective Least Slack Time First) can improve performance by including physical factor (moving time). As a future work, we plan to perform extensive simulations to verify performance of H-ELST in more realistic environment. In addition, we will study various algorithms for CPS considering socio factors.

**ACKNOWLEDGEMENTS**

This research is supported by 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, and L.E. Kavraki, Real-Time Guidance Under MRI, Engineering in Medicine and Biology Magazine, IEEE, Vol. 29, No. 2, 2010, pp. 78~86.

[2] C. Domínguez, H. Hassan and A. Crespo, Real-time Embedded Architecture for improving Pervasive Robot Services, International Journal of Software Engineering and Its Application. Vol. 2, No 1, 2008, pp. 79~90.

[3] S-H Park, J-H Kim, C-H Han, and K-S Kim, Effective One-to-One Correspondence Method of O(N^2 Log(N)) Complexity between Distributed Units, International Journal of Advanced Robotic Systems, Vol. 9, 2012, pp. 1~7.

[4] S-H Park, J-H Kim, and C-H Han, The Extended-Military Multimedia Systems Based on Real-Time Scheduling Scheme, Journal of The Institute of Electronics Engineers of Korea, Vol. 48-CI, No 1, 2011, pp. 26~32.

[5] C-H Han, Y-H Min, S-H Park, and J-H Kim, Minimum-Cost Path Finding Algorithm in Real-Time For Computer Generated Force, Journal of The Institute of Electronics Engineers of Korea, Vol. 48-CI, No 1, 2011, pp. 17~25.

[6] S-H Park, and J-H Kim, Real-time NCW Systems Using Distributed Processing, Kisse, Korea Computer Congress, Vol. 36, 2009, pp. 245-249.

[7] W.H. Yuan and K. Nahrstedt, Energy-efficient soft real-time CPU scheduling for mobile multimedia systems, 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, Vol. 27, 2011, pp. 45~58.

[9] J. Wang, S. Han, K-Y Lam and K. Mok, Maintaining data temporal consistency in distributed real-time systems, Real-Time Systems, Vol 48, No. 4, 2012, pp. 387~429.

[10] Q.Y. Dai1, R.Y. Zhong1, M.L. Wang, X.D. Liu and Q. Liu, RFID-enable Real-time Multi-experiment Training Center Management System, International Journal of Advanced Science and Technology. Vol. 7, 2009, pp. 27 ~48.

[11] C.L. Liu, J. Layland, Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, Journal of the ACM. Vol. 20, 1973, pp. 46~61.

[12] A. Burns, Scheduling hard real-time systems: a review, Software Engineering Journal. Vol. 6, 1991, pp. 116~128.

[13] Z.R.M. Azmi, K.A. Bakar, M.S. Shamsir, W.N.W. Manan and 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, Vol. 4, No.3, 2011, pp. 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, DAC '10 Proceedings of the 47th Design Automation Conference, Jun 13-18; Anaheim, California. USA, 2010, pp. 737~742.

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

[17] J.C. Eidson, E.A. Lee, S. Matic, S.A. Seshia and J. Zou. Distributed Real-Time Software for Cyber–Physical Systems, IEEE Proceedings, Vol. 100, 2011, pp. 45~59.

[18] E.A. Lee. Computing needs time, Communications of the ACM. Vol 52, No. 5, 2009, pp. 70~79.

[19] A. Banerjee, Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber–Physical Systems, Proceedings of the IEEE, Vol. 100, No. 1, 2012, pp. 283~299.

[20] J-H Kim, S-H Park, and Geoffery Fox, Real-Time Scheduling in Cyber-Physical Systems, International Conference, CCA, 2012, P. 69.

[21] Hai Zhuge, Cyber physical society, sixth international conference on semantics, knowledge and grids, 2010, pp. 1~8.

[22] Hai Zhuge, Semantic linking through spaces for cyber-physical-socio intelligence: A methodology, Artificial Intelligence, Vol. 175, 2011, pp. 988~1019.

[23] Hai Zhuge and Yunpeng Xing, Probabilistic resource space model for managing resources in cyber-physical society, IEEE transactions on services computing, VOL. 5, NO. 3, 2012, pp. 404~421.

[24] Ayman Faza, Sahra Sedigh, and Bruce McMillin, Integrated Cyber-Physical Fault Injection for Reliability Analysis of the Smart Grid, Proceedings of the 4th Annual ISC Research Symposium ISCRS 2010, April 21, Rolla, Missouri SAFECOMP, LNCS 6351, 2010, pp. 277~290.

[25] Ajinkya Bhave, Bruce H. Krogh, David Garlan, and Bradley Schmerl, View Consistency in Architectures for Cyber-Physical Systems, IEEE/ACM 2nd. Int. Conf. on Cyber-Physical Systems, April, 2011, pp. 151~162.

[26] T Yoshimoto, and T Ushio, Poster Abstract: Design of Modified Observer to Reduce State Estimation Error Caused by Job Skipping in Cyber-Physical Systems., 2012 IEEE/ACM Third International Conference, 2012, p 232.

[27] H.Zhuge, The Knowledge Grid: Toward Cyber-Physical Society, World Scientific Publishing Co., Singapore, 2012. 2nd Edition.

[28] Tiziana Ferrari and Luciano Gaido, Resources and Services of the EGEE Production Infrastructure, J. of Grid Computing, Volume 9, Issue 2, June 2011, pp. 119-133.

[29] Antun Balaž et al., Development of Grid e-Infrastructure in South-Eastern Europe, J. of Grid Computing, Volume 9, Issue 2, June 2011, pp. 135-154.

[30] Francisco Brasileiro et al., Using a Simple Prioritisation Mechanism to Effectively Interoperate Service and Opportunistic Grids in the EELA-2 e-Infrastructure, J. of Grid Computing, Volume 9, Issue 2, June 2011, pp. 241-257.

[31] Bryan Che, Red Hat Enterprise MRG: Messaging, Realtime, and Grid, [www.redhat.com](http://www.redhat.com)

[32] [Sangho Yi](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Sangho%20Yi.QT.&searchWithin=p_Author_Ids:37531687600&newsearch=true) et al., Towards Real-Time, Volunteer Distributed Computing, 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2011, pp. 154 – 163.