Dual-mode r-reliable Task Model for Flexible Scheduling in Reliable Real-time Systems

Kyong Hoon Kim, Jong Kim, and Sung Je Hong

Department of Computer Science and Engineering
Pohang University of Science and Technology (POSTECH)
Pohang, Korea
{jysh, jkim, sjhong}@postech.ac.kr

Abstract. Recent research in real-time systems has much focused on new task models for flexible scheduling and fault-tolerant real-time systems. In this paper, we propose a novel task model for the purpose of flexible scheduling in reliable real-time systems. In the proposed dual-mode r-reliable task model, a task periodically releases fast mode jobs or reliable mode jobs with the constraint that reliable mode jobs must be executed at least once for any r consecutive periods to guarantee the reliability of task. We also propose scheduling algorithms and compare performance through simulation results.

1 Introduction

Real-time systems are the systems in which the completion time of task affects systems as well as the correct result of task. Traditional real-time systems are classified into hard and soft real-time systems. In hard real-time systems, missing task deadlines is fatal for the task. By contrast, in soft real-time systems, a few missed deadlines do not cause serious problems, but the task has some diminished value for the failure of deadlines. Recent research in real-time systems, however, has focused on new task models for the purpose of the flexibility of task scheduling or the reliability of systems.

For more flexible scheduling, (m, k)-*firm* task model [1], *s-skippable* task model [2], and window-constrained task model [3] were proposed. Bernat et al. [4] generalized those task models and introduced *weakly hard* real-time systems. Weakly hard real-time systems are hard but permit occasional misses of task deadlines, so that tasks have their own bounded numbers to tolerate missed deadlines during any consecutive window of time. Moreover, Mok and Chen [5] proposed the *multiframe* task model, which allows the execution time of a task to vary from one instance to another with a known pattern. Another flexible task models are Imprecise Computation (IC) [6, 7] and Increased-Reward-with-Increased-Service (IRIS) [8]. A task in IC model consists of a mandatory part and an optional part. In IRIS model, each task receives a reward that depends on the amount of service received prior to deadline.

In addition to flexible scheduling, much research on fault-tolerant real-time systems has been conducted. In critical real-time systems, tasks must meet their

deadlines in spite of some failures. Fault-tolerant real-time scheduling algorithms have been proposed with both *forward recovery* and *backward recovery* schemes. For forward recovery in fault-tolerant real-time systems, the scheduling algorithms in [9–11] considered task replications and Ghosh et al. [12] analyzed the Rate Monotonic scheduling with task duplications. Punnekkat et al. [13] analyzed checkpointing for backward recovery of fault-tolerant real-time systems.

There have been recent works related on flexible and reliable real-time control systems. The Simplex architecture [14, 15] is a flexible and reliable software architecture that uses analytically redundant controllers. In the Simplex architecture, a device is controlled by two different versions of programs, a high-assurance controller and a high-performance controller. The high-assurance controller provides high reliability with low performance. The high-performance controller, however, shows good performance but low reliability. Chandra et al. [16] presented a method that finds the optimal frequencies for systems using analytically redundant controllers on the assumption that the failure rates are known.

In this paper, we introduce a new task model in which a task must execute at least one reliable job during the bounded window of time. A task periodically releases jobs which may be in *fast* mode or *reliable* mode. The fast mode job is the normal execution process to accomplish the goal of task, while the reliable mode job adds more dependable process for the reliability of task. In wireless real-time communication, for instance, reliable job data may be sent with a large amount of redundant bits to tolerate packet loss, while fast mode job data are sent with few redundant bits. Besides two different job modes, we specify the minimum number of reliable mode jobs to be executed. A task with the bound of r must execute at least one reliable job for any r consecutive jobs, which will be defined by *r-reliable constraint*. Thus, the proposed task model is named the *dual-mode r-reliable* task model. The applications of the proposed task model include reliable real-time communication, fault-tolerant real-time scheduling, and QoS or multimedia related applications. This paper will also suggest several priority-driven scheduling algorithms for the proposed task model.

The remainder of this paper is organized as follows. In Section 2, we specify the new task model and discuss the scheduling problem. The scheduling algorithms for the proposed task model are described in Section 3 and simulation results of those algorithms are given in Section 4. Finally, this paper is concluded with Section 5.

2 Dual-mode r-reliable Real-time Systems

The dual-mode r-reliable real-time system is a real-time system which consists of the dual-mode r-reliable tasks. This section defines dual-mode r-reliable task model and the scheduling problem to solve.

2.1 Dual-mode r-reliable Task Model

A dual-mode r-reliable real-time task can be explained by three terminologies: *real-time, dual-mode*, and *r-reliable*. Since we assume a task is periodic, a *real-*

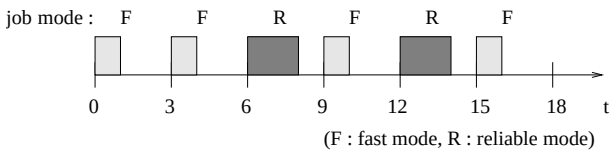

Fig. 1. Example of a dual-mode r-reliable task $\tau(2, 1, 3, 3)$

time task means that the task periodically creates jobs which must be finished before the next job releases. *Dual-mode* means that the computation mode of each job is either in $fast$ mode or *reliable* mode. Lastly, *r-reliable* notifies the constraint that at least one job of every r consecutive jobs of the task is in reliable mode. Thus, a dual-mode r-reliable real-time task τ is defined by four parameters (c^R, c^F, p, r).

- c^R : the execution time of reliable mode job
- c^F : the execution time of fast mode job
- p : the period of jobs released
- r : r-reliable constraint

A task τ releases a job every p time units. Each job of the task can be executed as soon as it is released and its relative deadline is p. As stated above, a job is either in fast mode or reliable mode. According to the computation mode, the required execution time of the job is c^F or c^R in each. In general, the reliable mode job requires more processing time so that we assume the execution time of the reliable mode job is larger than that of the fast mode job ($c^R > c^F$). The parameter r indicates how often the reliable mode job should be executed for the reliability of the task. Thus, we represent the reliability of a task with this *r-reliable constraint*.

For example, the task $\tau(2, 1, 3, 3)$ can be scheduled as shown in Figure 1. The task creates a job every 3 time units, and the job is either in fast mode or reliable mode. While the fast mode job is executed for one time unit, the reliable mode job is executed for two time units. Since at least one reliable job is executed during every three consecutive periods, the schedule in Figure 1 meets the 3-reliable constraint.

The reliable mode job of a task can be modeled in two different manners. The first is an additional computation which is independent of the fast mode job and requires ($c^R - c^F$) computation times. The second approach of the reliable mode job is to execute more reliable process different from the fast mode job. One such example is the MPEG decoding process, in which I-frame in MPEG decoding differs from P-frame decoding and takes more time. Reliable real-time communications using FEC (Forward Error Correct) schemes are also an example. A low rate-coded massage with less redundant information corresponds to the fast mode job, while a high rate-coded message with more redundant information corresponds to the reliable mode job. In this paper, we assume the latter case so that the computation process of the reliable mode job is different from that of the fast mode job.

The main applications of dual-mode r-reliable task model are reliable real-time communication, fault-tolerant real-time scheduling, and QoS or multimedia related applications. In reliable real-time communication, it is necessary to send data with more redundant bits to tolerate packet loss but we cannot send such data every time for the lack of network bandwidth. These data can be represented with a reliable mode job in the proposed task model and its minimum distance corresponds to the r-reliable constraint. We may apply the model to fault-tolerant real-time scheduling using backward error recovery scheme with checkpointing. That is, a normal job added with checkpointing process is modeled as a reliable mode job and the checkpointing period indicates the r-reliable constraint. Moreover, many real-time applications related on QoS or multimedia can use the r parameter in such a manner that tasks with higher qualities are assigned with lower r's. For example, the standard H.261 and H.263 ITU video codecs have two different frame types, Intraframe (I-frame) and Interframe (P-frame). Data size of P-frame is smaller than that of I-frame since it is based on the previous frame (I-frame or P-frame). Thus, both H.261 and H.263 bitstreams are modeled with dual-mode r-reliable tasks and the minimum interval between I-frames corresponds to th r-reliable constraint.

The proposed dual-mode r-reliable task can be modeled with previous task models [17, 5, 1]. A dual-mode r-reliable task can be modeled as the task with the worst-case execution time c^R in the L&L task model [17], or the multiframe task with one reliable job and $(r-1)$ consecutive fast mode jobs in the multiframe task model [5]. However, these are so pessimistic that their schedulabilities are low. The $(m,\ k)$-firm task model [1] can also be used for modeling a dual-mode r-reliable task by decomposing the task into (1, 1)-firm task with execution time c^F and (1, r)-firm task with execution time $(c^R - c^F)$. It is only suitable for the case that the reliable mode job is an additional process independent of the fast mode job. However, as we assume the reliable mode job is different process from the fast mode job, the $(m,\ k)$-firm task model does not exactly model the proposed task.

2.2 Scheduling Problem

Before turning to the scheduling problem, we address the feasibility. A schedule of a given dual-mode r-reliable task set is *feasible* if the schedule makes all jobs meet their deadlines and each task meet its r-reliable constraint. Also, a dual-mode r-reliable task set T is said to be *feasible* if and only if there exists a scheduling algorithm to generate a feasible schedule of the task set.

Quan and Hu [18] proved that the scheduling problem of the (m, k)-firm task model is NP-hard. In the assumption that all the computation times of fast mode jobs are zero, the scheduling problem of dual-mode r-reliable tasks is equal to that of corresponding $(1, r)$-firm tasks. Thus, the scheduling problem of the dual-mode r-reliable task model is NP-hard in the weak sense.

The *minimum effective utilization* of a dual-mode r-reliable task $\tau(c^R, c^F, p, r)$ is defined by $\frac{c^F}{p} + \frac{c^R - c^F}{p \times r}$. The minimum effective utilization of the task τ is the

minimum fraction of time that the task τ keeps a processor busy, since τ requires c^F time units every p periods and at least $(c^R - c^F)$ times are allocated to τ's reliable jobs for $p \times r$ time units. The minimum effective utilization $U^e(T)$ of a dual-mode r-reliable task set T is the summation of minimum effective utilization of the individual tasks within it.

Lemma 1. *Given a dual-mode r-reliable task set* $T = \{\tau_i = (c_i^R, c_i^F, p_i, r_i) | 1 \leq i \leq n\}$, *if* $U^e(T)$ *is larger than 1, then* T *is not feasible.*

Proof. Even if we fully utilize the processor to execute the task set, the utilization is 1. Since $U^e(T) > 1$, a task exists which cannot be scheduled. Thus, the task set T is not feasible. □

The task set satisfying the condition in Lemma 1 cannot be scheduled by any algorithm due to its over-utilization. By contrast, if the utilization of a task set is small enough to schedule every job as a reliable mode job, the task set is always schedulable. The following lemma tells the condition to guarantee that every task in the task set is schedulable.

Lemma 2. *Given a dual-mode r-reliable task set* $T = \{\tau_i = (c_i^R, c_i^F, p_i, r_i) | 1 \leq i \leq n\}$, T *is feasible if* T *satisfies* $\sum_{\tau_i \in T} \frac{c_i^R}{p_i} \leq 1$.

Proof. Consider the EDF algorithm and assume all tasks always release their jobs in the reliable mode. Since the EDF is optimal, all reliable mode jobs of tasks are schedulable if $\sum_{i=1}^{n} \frac{c_i^R}{p_i} \leq 1$. Furthermore, all jobs meet their r-reliable constraints. Thus, T is feasible. □

By Lemmas 1 and 2, scheduling algorithms in the next section concentrate on scheduling the task set T such that $U^e(T) \leq 1$ and $\sum_{\tau_i \in T}(c_i^R/p_i) > 1$.

3 Scheduling Algorithms

In this section, we propose one fixed priority-based algorithm and three dynamic priority-based algorithms for dual-mode r-reliable real-time systems. The fixed priority-based algorithm described in Section 3.1 is based on the RM, and dynamic priority-based algorithms in Section 3.2 are based on the EDF.

3.1 Fixed Priority-based Algorithm

DR-RM (Dual-mode r-Reliable Rate Monotonic) assigns priorities to tasks according to the Rate Monotonic policy [17]: the shorter period, the higher priority. In DR-RM, the reliable mode job pattern of the task with r-reliable constraint is determined in such a manner that every r-th job is in reliable mode. For instance, the job mode pattern of a 4-reliable task is FFFRFFFR$\cdots$ (F : fast mode, R : reliable mode).

As we fix the reliable mode job pattern of a task, a dual-mode r-reliable task $\tau(c^R, c^F, p, r)$ is modeled as the multiframe [5] task with period p and r

execution time list which consists of $(r-1)$ c^F's and one c^R. The *critical instance test* for testing the schedulability of multiframe tasks [5] is applicable to test the schedulability of dual-mode r-reliable tasks. Therefore, Theorem 1 follows.

Theorem 1. *Suppose that a dual-mode r-reliable task set $T = \{\tau_i(c_i^R, c_i^F, p_i, r_i) | 1 \leq i \leq n\}$ is given and tasks are sorted such that $p_1 \leq p_2 \leq \ldots \leq p_n$. Let us define the worst-case execution time $W_i(t)$ as follows:*

$$W_i(t) = c_i^R + \sum_{j=1}^{i-1} \left(\left\lceil \frac{t}{p_j} \right\rceil \cdot c_j^F + \left\lceil \frac{t}{p_j \times r_j} \right\rceil \cdot (c_j^R - c_j^F) \right)$$

If, for all $1 \leq i \leq n$, some $t \leq p_i$ exists such that $W_i(t) \leq t$, then all the tasks in the task set T are schedulable by the DR-RM and their r-reliable constraints are met.

Proof. Let us define $R_j(t)$ be $\left\lceil \frac{t}{p_j} \right\rceil \cdot c_j^F + \left\lceil \frac{t}{p_j \times r_j} \right\rceil \cdot (c_j^R - c_j^F)$ so that $W_i(t)$ is the sum of c_i^R and $\sum_{j=1}^{i-1} R_j(t)$. Consider the reliable mode job of task τ_i. The critical instance of task τ_i is when all reliable mode jobs of higher-priority tasks than τ_i are released at the same time. Then, $R_j(t)$ is the computation times of task τ_j until t because task τ_j requires c_j^F times for every p_j times and additional $c_j^R - c_j^F$ times for every $p_j \times r_j$ times. Thus, $W_i(t)$ becomes the sum of the computation times of the reliable mode job of task τ_i and the computation times of all jobs with higher-priority than τ_i at time t.

Therefore, if $t < p_i$ exists such that $W_i(t) \leq t$, then the reliable mode job of τ_i will meet its deadline. Since the reliable mode job of task τ_i meets its deadline, fast mode jobs also meet their deadlines. Moreover, DR-RM generates every r_i-th job with a reliable mode job so that the r_i-reliable constraint is met. Hence, task τ_i is schedulable and the theorem follows since i can be any number. □

3.2 Dynamic Priority-based Algorithms

Dynamic priority-based algorithms do not fix priorities of tasks so that the scheduler always executes the highest-priority job among available jobs. In this paper, we select the EDF policy for assigning the priorities of jobs. Thus, the earlier the deadline of the job, the higher the priority. Figure 2 shows the scheduling algorithm based on the dynamic priority scheme.

In Figure 2, the algorithm **Queue-Schedule** executes the job with the earliest absolute deadline in the *wait_queue* and the algorithm **Job-Release** releases every job among given tasks. At the time of releasing a job, the job mode is determined by the function *Determine_Mode*. In this paper, three different algorithms are used to determine job mode patterns. The following subsections describe those algorithms.

(1) FIX Algorithm

The FIX algorithm determines the job mode patterns in the same manner as that of DR-RM. That is, every r-th job is in reliable mode and others are in fast mode.

Algorithm Scheduling_Dual-mode_r-reliable
/* - t : current time
- *wait_queue* : the queue of current available jobs
*/
Concurrently execute **Queue-Schedule** and **Job-Release**.

Queue-Schedule
Execute the job which has the earliest absolute deadline in *wait_queue*.

Job-Release
for each task τ_i **do**
  if t mod p_i == 0 **then**
    mode = *Determine_Mode* (τ_i)
    if mode == RELIABLE **then**
      Create a reliable mode job of τ_i and insert it in *wait_queue*.
    else /* FAST mode */
      Create a fast mode job of τ_i and insert it in *wait_queue*.
    endif
  endif
endfor

Fig. 2. Dynamic priority-based algorithm

(2) DBP Algorithm

We modify the DBP (Distance-Based Priority) assignment scheme [1] to generate computation modes of tasks. Each task maintains $state_i$ which indicates the state of task τ_i in the state transition diagram of the DBP assignment [1]. The state transition diagram of a dual-mode r-reliable task is shown in Figure 3. At every state, when the task releases the reliable mode job, its state becomes $r-1$. When the r-reliable task of state j ($j > 0$) releases the fast mode job, its state becomes $j-1$. If the task of state 0 releases the fast mode job, its state remains at state 0, and at that time the task cannot meet the r-reliable constraint.

The DBP job mode determining algorithm assigns the task τ_i of the lowest $state_i$ to be in reliable mode. To prevent tasks from missing the r-reliable constraint, the task in state 0 always releases the reliable mode job.

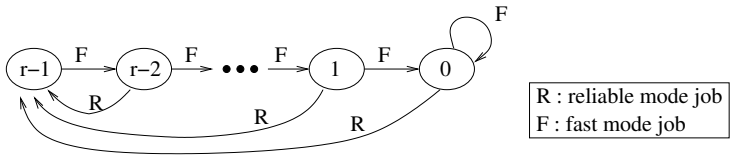

Fig. 3. The state transition diagram of r-reliable task

(3) SS Algorithm

The SS (Slack Stealing) algorithm for determining the job mode is based on generic slack stealing techniques for scheduling aperiodic and sporadic jobs. This paper references the work of Chetto and Chetto [19].

For a given dual-mode r-reliable task set T, we define e_i and Δ_i. e_i is the distinct i-th arrival time of jobs in system and Δ_i is the length of idle time between e_{i-1} and e_i under the EDF schedule of tasks in T assuming that all jobs are in fast mode. Chetto and Chetto [19] derive the equation for calculating Δ_i without scheduling the EDF until time e_i. Since we schedule all jobs of tasks with fast modes, the length of the idle time between e_{i-1} and e_i is given by the following equation.

$$\Delta_i = \max\{0,\ e_i - \sum_{j=1}^{n} \lceil \frac{e_i}{p_j} \rceil c_j^F - \sum_{k=1}^{i-1} \Delta_k\}, \qquad i = 1, 2, \ldots$$

The SS algorithm manages $dist\_reliable_i$ to indicate the distance to a reliable mode job. When $dis\_reliable_i$ is non-zero, the task τ_i creates a fast mode job and $dis\_reliable_i$ decreases by one. When $dis\_reliable_i$ is zero, the task τ_i releases a reliable mode job and then decides the next $dist\_reliable_i$. The brief algorithm to determine the next $dist\_reliable_i$ is finding the last-fit period in which enough idle times for the reliable mode job exists among the next r_i periods. If the summation of Δ_i's during each period is larger than or equal to $c_i^R - c_i^F$, then we can schedule the reliable mode job at that period.

4 Simulation Results

In this section, we present simulation results to compare the proposed algorithms in dual-mode r-reliable real-time systems. The simulation in this section is performed as follows: we randomly generate 500 feasible task sets for each minimum effective utilization bound. Each task set consists of at least five tasks. To generate a feasible task set, we start the empty task set and add a randomly generated task to the task set one by one. The randomly generated task is examined to test for schedulability, including tasks in the task set so that only the task to succeed in the schedulability test is added to the task set. We perform the schedulability test for the task by scheduling it with tasks in the task set until the least common multiple of $r_i \times p_i$'s of all tasks.

In the experiment of Figure 4, the period p_i of each task is given in the range from 4 to 25 and the reliable constraint r_i is in the range from 2 to 10. For the computation time, we randomly select c_i^R between 3 and p_i and then randomly generate c_i^F in the range from 2 to $c_i^R - 1$. The characteristics of feasible task sets in the experiment is shown in Table 1. Since dynamic priority-based algorithms proposed in the previous section are all based on the EDF policy, they are named FIX-EDF, DBP-EDF, and SS-EDF according to the job mode pattern algorithm.

Table 1. Characteristics of feasible task sets

Effective Util.	$0.4 \sim 0.5$	$0.5 \sim 0.6$	$0.6 \sim 0.7$	$0.7 \sim 0.8$	$0.8 \sim 0.9$
# of tasks / set	5.0	5.0	5.0	5.0	5.0
average c^R	4.5	4.8	5.1	5.0	4.7
average c^F	2.1	2.3	2.5	2.7	2.7
average p	21.2	20.1	19.4	18.5	17.6
average r	6.8	6.6	6.3	6.2	6.0

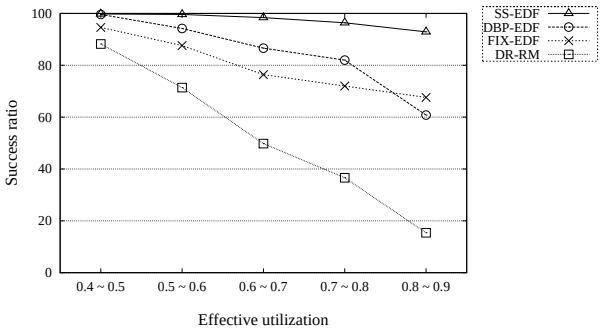

Fig. 4. The success ratios of algorithms

Figure 4 shows the success ratio of scheduling dual-mode r-reliable task sets in percentile. The success ratios of dynamic priority-based algorithms are higher than that of fixed priority-based algorithm DR-RM. SS-EDF shows the highest success ratio among dynamic priority-based algorithms.

5 Conclusions

In this paper, we proposed a dual-mode r-reliable real-time task model, which is a novel task model for flexible scheduling in reliable real-time systems. A dual-mode r-reliable task has two different mode jobs and its reliable mode job must be executed at least once for any r consecutive periods. The task model can be applied to many recent real-time applications, including reliable real-time communication, fault-tolerant real-time systems, and QoS related applications. We also suggested priority-driven scheduling algorithms for the proposed task model and compared performance throughout simulation results.

Acknowledgement

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2005-C1090-0501-0018).

References

1. Hamdaoui, M., Ramanathan, P.: A dynamic priority assignment technique for streams with (m,k)-firm deadlines. IEEE Transactions on Computers **44(12)** (1995) 1443–1451
2. Koren, G., Shasha, D.: Skip-Over: algorithms and complexity for overloaded systems that allow skips. Proceedings of Real-Time Systems Symposium (1995) 110–117
3. West, R., Poellabauer, C.: Analysis of a window-constrained scheduler for real-time and best-effort packet streams. Proceedings of Real-Time Systems Symposium (2000) 239–248
4. Bernat, G., Burns, A., Llamosi, A.: Weakly hard real-time systems. IEEE Transactions on Computers **50(4)** (2001) 308–321
5. Mok, A.K., Chen, D.: A multiframe model for real-time tasks. IEEE Transactions on Software Engineering **23(10)** (1997) 635–645
6. Lin, K.-J., Natarajan, S., Liu, J. W.-S.: Imprecise results: utilizing partial computations in real-time systems. Proceedings of Real-Time Systems Symposium (1987) 210–217
7. Liu, J. W.-S., Lin, K.-J., Shih, W.-K., Yu, A. C.-S., Chung, C., Yao, J., Zhao, W.: Algorithms for scheduling imprecise computations. IEEE Computers **24(5)** (1991) 58–68
8. Dey, J. K., Kurose, J., Towsley, D.: On-line scheduling policies for a class of IRIS (Increasing Reward with Increasing Service) real-time tasks. IEEE Transactions on Computers **45(7)** (1996) 802–813
9. Liestman, A. L., Campbell, R. H.: A fault-tolerant scheduling problem. IEEE Transactions on Software Engineering **12(11)** (1986) 1089–1095
10. Krishna, C., Shin, K.G.: On scheduling tasks with a quick recovery from failure. IEEE Transactions on Computers **35(5)** (1986) 448–455
11. Han, C.-C., Shin, K.G., Wu, J.: A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults IEEE Transactions on Computers **52(3)** (2003) 362–372
12. Ghosh, S., Melhem, R., Mosśe, D., Sarma, J. S.: Fault-tolerant rate-monotonic scheduling. Journal of Real-Time Systems **15(2)** (1998) 149–181
13. Punnekkat, S., Burns, A., Davis, R.: Analysis of checkpointing for real-time systems. Journal of Real-Time Systems **20(1)** (2001) 83–102
14. Seto, D., Krogh, B., Sha, L., Chutinan, A.: Dynamic control systems upgrade using Simplex architecture. IEEE Control Systems (1998) 72–80
15. Sha, L.: Dependable system upgrade. Proceedings of Real-Time Systems Symposium (1998) 440–448
16. Chandra, R., Liu, X., Sha, L.: On the scheduling of flexible and reliable real-time control systems. Journal of Real-Time Systems **24(2)** (2003) 153–169
17. Liu, C.L., Layland, J.W.: Scheduling algorithms for multi-programming in a hard real-time environment. Journal of ACM **20** (1973) 46–61
18. Quan, G., Hu, X.: Enhanced fixed-priority scheduling with (m,k)-firm guarantee. Proceedings of Real-Time Systems Symposium (2000) 79–88
19. Chetto, H., Chetto, M.: Some results of the earliest deadline scheduling algorithm. IEEE Transactions on Software Engineering **15(10)** (1989) 1261–1269