symmetry

MDPI

Article

An N-Modular Redundancy Framework Incorporating Response-Time Analysis on Multiprocessor Platforms

Jaemin Baek [1], Jeonghyun Baek [2], Jeeheon Yoo [3] and Hyeongboo Baek [3,*]

[1] Agency for Defense Development (ADD), Defense Satellite Systems PMO, Daejeon 34063, Korea
[2] Agency for Defense Development (ADD), The 3rd R&D Institute-3rd Directorate, Daejeon 34063, Korea
[3] Department of Computer Science and Engineering, Incheon National University (INU), Incheon 22012, Korea
* Correspondence: hbbaek@inu.ac.kr; Tel.: +82-32-835-8493

Received: 8 July 2019; Accepted: 27 July 2019; Published: 31 July 2019

Abstract: A timing constraint and a high level of reliability are the fundamental requirements for designing hard real-time systems. To support both requirements, the N modular redundancy (NMR) technique as a fault-tolerant real-time scheduling has been proposed, which executes identical copies for each task simultaneously on multiprocessor platforms, and a single correct one is voted on, if any. However, this technique can compromise the schedulability of the target system during improving reliability because it produces N identical copies of each job that execute in parallel on multiprocessor platforms, and some tasks may miss their deadlines due to the enlarged computing power required for completing their executions. In this paper, we propose task-level N modular redundancy (TL-NMR), which improves the system reliability of the target system of which tasks are scheduled by any fixed-priority (FP) scheduling without schedulability loss. Based on experimental results, we demonstrate that TL-NMR maintains the schedulability, while significantly improving average system safety compared to the existing NMR.

Keywords: hard real-time systems; schedulability; reliability; N modular redundancy; multiprocessor platform; fixed-priority scheduling

1. Introduction

Most embedded systems are known as real-time, of which timing constraints are one of the system design requirements. One of the most important timing constraints in real-time systems is to complete executions within their corresponding deadlines [1]. Therefore, the correctness of a real-time system depends on the time in which the correct logical and functional output is generated. One class of real-time systems with strict timing constraints is the hard real-time system. If the timing constraints of hard real-time systems are not met, the consequences can be life threatening or cause serious economic losses. Therefore, hard real-time system designers must ensure that all timing constraints are met before the target system is actually implemented.

In addition to meeting the timing constraints of a hard real-time system, the systems functional accuracy should be guaranteed [2]. An output produced in a timely manner is not reliable when the system is out of the specified functional correctness. This deviation occurs in a computer system when the system is defective. Therefore, hard real-time systems must meet all timing constraints of tasks while they are not defective. Timing constraints in real-time applications can be met through proper job scheduling, and fault tolerance can be achieved by obtaining the required level of reliability.

Although many fault-tolerant techniques have been previously implemented using hardware[3], recent software-based techniques such as rollback, checkpointing, and re-execution have been proposed, which were initially designed for uniprocessor systems [4–6]. Checkpoints using the

rollback technology manage each checkpoint, where the state of the system is stored in stable storage, and the system status is restored to the latest checkpoint if a transient error is detected [4]. The re-execution technique runs a job multiple times and selects the correct output obtained during multiple runs [5,6]. If all outputs are incorrect during the multiple runs, it re-executes the job to improve system reliability. Under the re-execution technique, tasks execute multiple times, which increases the possibility of deadline misses due to prolonged execution times. Therefore, existing research considering the re-execution technique aims at improving the system reliability of mixed-critical systems or energy-sensitive real-time systems; however, the schedulability of the system has inevitably been compromised.

In multi-processor domains, errors can be tolerated by taking advantage of the functionality of multiple processors [7–9]. The most common approach is the primary-backup approach in which the backup of a task executes if its primary does not execute successfully [7]. Backup overloading approach schedules a backup copy of the primary job in a time-overlapping manner for operational efficiency [8]. Another efficient overloading algorithm on multiple processors has been proposed through dynamic logical grouping between copies of tasks [9]. Regarding a hardware approach, Cirinei et al. proposed a dynamic reconfiguration of a multiprocessor hardware platform with a balance between performance and fault tolerance through concurrent replication [10]. N modular redundancy (NMR) executes identical copies for each task simultaneously on multiprocessor platforms, and a single correct output is voted if any. Because this technique produces N identical copies of each job, which execute in parallel, some tasks may miss their deadlines owing to enlarged computing power required for completing their execution [4–6].

Although NMR to make the target system tolerant to a transient fault is an effective approach for real-time scheduling, it can compromise the schedulability of the target system while improving reliability [4]. Because this technique produces N identical copies of each job, which are executed in parallel, some tasks may miss their deadlines owing to enlarged computing power required for completing their executions. This is due to the N modular redundancy techniques limited capability of forcing the same N number of copies for all tasks, where the number N is determined without schedulability analysis [4,6].

In this study, we propose a task-level N modular redundancy (TL-NMR) technique, which improves the system reliability of the target system in which tasks are scheduled by any fixed-priority (FP) scheduling without schedulability loss. The TL-NMR framework determines the number of copies (not the same number of copies for all the tasks) of each job J_k^q of a task τ_k while ensuring that every copied job can complete its execution before its absolute deadline by effectively using a new response-time analysis (RTA) proposed in this paper. Then, N_k copies of each job J_k^q execute simultaneously on multiple processors under the given FP scheduler, and a single correct output (if any) is voted on. Based on the experimental results, we demonstrate that TL-NMR maintains schedulability while significantly improving the average system safety, compared to the existing NMR.

The remainder of this paper is organized as follows. Section 2 presents our system model including task and reliability models. Section 3 introduces our proposed fault-tolerant scheduling framework called TL-NMR. Section 5 evaluates RTA for TL-NMR with various performance metrics. Section 6 concludes this study.

2. System Model

In this section, we present the system model of our target system, which includes task and reliability models.

2.1. Task Model

We consider the Liu and Layland task model [1] for m identical processors in a hard real-time system, in which a set τ of tasks $\tau_k (1 \leq k \leq |\tau|)$ are denoted by three tuples of parameters, i.e., $\tau_k = (T_k, C_k, D_k)$. For a given period T_k, worst-case execution time C_k, and relative deadline D_k, a task τ_k is

supposed to invoke a series of jobs whose arrivals (also called release times) are at least T_k time units away from each other. In addition, each job of τ_k is required to execute for at most C_k time units to complete its execution and such an execution should be ended within D_k time units.

The q-th job J_k^q is invoked by a task τ_k at its release time r_k^q, and its execution should be completed before its absolute deadline d_k^q as a timing constraint. We let the q-th job J_k^q's finishing time be f_k^q, i.e., f_k^q should be less than or equal to d_k^q to satisfy the timing constraint; then the job J_k^q is said to be schedulable. Consequently, a task τ_k is schedulable if every job of τ_k is schedulable, and a task set τ is schedulable when every task τ_k is schedulable. When the TL-NMR technique is applied, N_k copies are produced for each job J_k^q. Here, each copy (i.e., each copied job) is denoted by $J_k^{q,p}$, where $1 \leq p \leq N_k$ holds, and a task τ_k is schedulable if every job $J_k^{q,p}$ of τ_k is schedulable. We assume that each single job cannot execute in parallel on multiple processors simultaneously. All copied jobs $J_k^{q,p}$ can have different finishing times $f_k^{q,p}$ depending on the scheduling algorithm; however, they have the same release time r_k^q and absolute deadline d_k^q.

The response time R_k of a task τ_k is given by $\max_{J_k^q \in \tau}(f_k^q - r_k^q)$ ($max_{J_k^{q,p} \in \tau}(f_k^{q,p} - r_k^q)$ when the TL-NMR technique is applied). By the definition of the response time R_k, a task set τ is guaranteed to be schedulable if $R_i \leq D_i$ holds. We target a constrained-deadline task system in which $C_k \leq D_k \leq T_k$ holds for every task $\tau_k \in \tau$. We consider a global preemptive work-conserving scheduling algorithm in which a job can migrate from one processor to another and a lower priority job's execution can be hindered by a higher priority one. We assume quantum-based time where a time unit describes a quantum length of one.

2.2. Reliability Model

Among various types of faults, we consider the transient fault for our system model, which occurs for a short time without damaging the hardware devices. Reliability is the degree of tolerance for target systems against transient faults, which has been addressed by a number of existing studies, regarding a task (called task reliability) and a system (called system reliability) [6]. Task reliability is defined as the possibility that a single job of the task can execute without a transient fault. For a given average fault, arrival rate γ and exponential distribution, the task reliability Y_k of a task τ_k is given by [6].

$$Y_k = e^{-\gamma C_k}. \tag{1}$$

We assume that at most one transient fault can occur for a single job and it does not change the worst-case execution time C_k of a task τ_k as a common assumption [11].

In the existing NMR, each job's N identical copies are executed on multiple processors simultaneously (sharing the same release time $r_k^{q,p}$ and absolute deadline $d_k^{q,p}$), and their results are voted to produce a single correct output (with no transient faults) if any. Unlike the existing NMR technique, our proposed TL-NMR allows different tasks to execute different number of copies for each execution. We denote the number of such copies by N_k (each τ_k can have different values of N_k). We describe the method to determine N_k in Section 3. By the definition of reliability, $1 - Y_k$ implies the possibility that a job J_k^q does not successfully execute owing to a transient fault. In addition, TL-NMR selects a correct job (if any) among N_k identical jobs. Thus, Y_k under the TL-NMR technique can be re-formulated as follows:

$$Y_k = 1 - (1 - e^{-\gamma C_k})^{N_k}. \tag{2}$$

Then, system reliability $Y(\tau)$ is defined as the average of task reliability Y_k in a task set τ and is calculated as

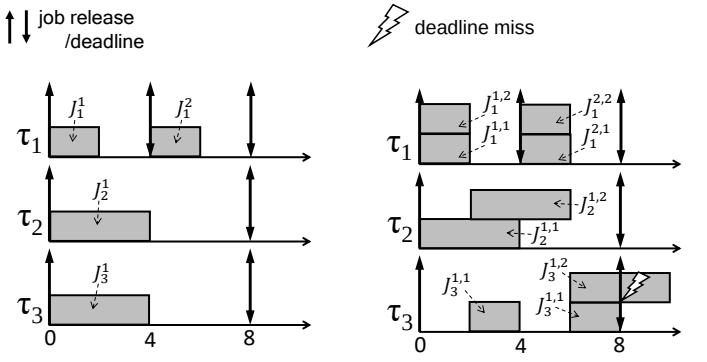
$$Y(\tau) = \frac{\sum_{\tau_k \in \tau} Y_k}{|\tau|} \tag{3}$$

For designing a hard real-time system, the reliability of the system should be maintained at a high level and every single execution of a task should be finished within its corresponding absolute deadline. The system safety $\Omega(\tau)$ to quantify system reliability and schedulability simultaneously is defined as the product of system reliability $\Upsilon(\tau)$ and schedulability (one if schedulable and zero otherwise) of τ. Thus, $\Omega(\tau)$ is $\Upsilon(\tau)$ if $R_k \leq D_k$ for all $\tau_k \in \tau$, and zero otherwise.

3. TL-NMR Framework

In this section, we propose TL-NMR, which improves the system reliability of a target system in which tasks are scheduled by any FP scheduling without schedulability loss. The TL-NMR framework determines N_k, which represents the individual number of copies of each job J_k^q of a task τ_k by effectively using a new RTA proposed in this paper. Then, N_k copies of each job J_k^q execute simultaneously on multiple processors under the given FP scheduler, and a single correct output (if any) is voted.

Because a real-time scheduling algorithm exploiting the existing NMR technique schedules N copies of each job, it requires N times C_k (i.e., $N \cdot C_k$) for each τ_k. Thus, some jobs (although they satisfy their absolute deadlines under vanilla (It indicates a scheduling algorithm that does not incorporate NMR.) FP scheduling) may miss their absolute deadlines owing to prolonged execution times. Figure 1 illustrates the scenario where a schedulable task set under vanilla RM scheduling RM scheduling (assigns a higher priority to that task τ_k which has a small T_k.) becomes unschedulable owing to NMR. τ_2 and τ_3 have the same priority owing to their same T_i; however, we assume that τ_2 has a higher priority than τ_3 according to a simple tie-breaking rule that a task with a smaller task index has a higher priority. As shown in Figure 1, jobs of three tasks τ_1, τ_2, and τ_3 are schedulable under vanilla RM on $m = 3$ processors (Figure 1a). Considering RM scheduling that exploits NMR for $N = 2$, two identical copies of each job are scheduled (Figure 1a). The first job $J_3^{1,1}$ of a job J_3^1 starts its execution at time instant $t = 2$, and it is preempted at $t = 4$ owing to higher priority jobs. Then, it completes its execution at $t = 8$. However, the second job $J_3^{1,2}$ of the job J_3^1 begins its execution at $t = 6$ because all the three processors are occupied by higher priority jobs. Then, it misses its absolute deadline at $t = 8$ because it does not execute completely for $C_k = 4$.



(**a**) Vanilla RM (**b**) RM incorporating NMR ($N = 2$)

Figure 1. Schedules under vanilla RM and RM incorporating NMR ($N = 2$) for an example task set $\tau = \{\tau_1(T_1 = 4, C_1 = 2, D_1 = 4), \tau_2 = \tau_3(8, 4, 8)\}$ on $m = 3$ processors.

We can easily observe that τ in Figure 1b becomes schedulable if one copied job of any task is not considered for scheduling (e.g., $N_k = 1$ for any τ_k). TL-NMR is capable of assigning N_k for each task, while it does not make a schedulable task set unschedulable under any FP scheduling. To achieve this goal, we need to address the following questions:

- How to determine N_k of each task τ_k?
- How to guarantee schedulability of τ_k for a given N_k?

To address question 1, TL-NMR effectively assigns the value of N_k using N_k-assignment algorithm in conjunction with a new RTA for TL-NMR so that R_k of every task is never greater than D_k. With N_k of every task τ_k, TL-NMR makes N_k identical copies of each job and schedules a task set τ according to the given FP scheduling algorithm. To address question 2, we develop a new RTA for TL-NMR to guarantee that there is no deadline miss while N_k identical copies of each job are scheduled by the given FP scheduling algorithm.

Algorithm 1 presents the operation of TL-NMR. At the beginning, N_k for every task τ_k is set to one. Then, N_k of each task is determined (Lines 2–8). For every task τ_k in τ, schedulability is tested with a value of $N_k + 1$ using RTA for TL-NMR, and $N_k + 1$ is assigned for τ_k if τ_k is deemed schedulable by the test (Lines 4–6). This procedure is conducted $m - 1$ times (Lines 2–8) because at most m copies of each job are allowed by TL-NMR on an m processor platform. Thereafter, a given task set τ is scheduled (Lines 9–17). For every time instant t, a job J_k^q of a task τ_k is inserted in a ready queue Q whenever J_k^q is released (Lines 9–11). Released jobs in Q are scheduled according to the base FP scheduling algorithm (Line 13). Finally, J_k^q is removed from Q when the execution of all the copied jobs $J_k^{q,p}$ of J_k^q is completed.

Algorithm 1 TL-NMR

1: $N_k \leftarrow 1$ for all tasks $\tau_k \in \tau$
2: **for** from 1 to $m - 1$ **do**
3: **for** τ_k in τ **do**
4: **if** τ is deemed schedulable with $N_k + 1$ by a given new RTA **then**
5: $N_k \leftarrow N_k + 1$
6: **end if**
7: **end for**
8: **end for**
9: **for** Every time instant t **do**
10: **if** J_k^q is released by τ_k **then**
11: Insert J_k^q into Q
12: **end if**
13: Schedule jobs (N_k copied jobs $J_k^{q,p}$ for each job J_k^q) in Q according to a given FP scheduling
14: **if** All copied jobs $J_k^{q,p}$ of J_k^q finish its execution **then**
15: Delete J_k^q from Q
16: **end if**
17: **end for**

4. New RTA for TL-NMR

Because our goal is to improve reliability while guaranteeing schedulability under TL-NMR, we should be able to judge whether the task set τ is schedulable with the given values of N_k for every task τ_k while N_k is assigned by TL-NMR. Hence, we develop a new RTA that can be incorporated into TL-NMR.

The response time R_k of a task τ_k can be upper-bounded by the summation of the worst case execution time C_k and the worst case time instance hindering the execution of a job J_k^q of τ_k. Because C_k is given, RTA for TL-NMR upper bounds the latter by exploiting the notion of interference in an interval $[r_k^q, r_k^q + \ell)$ (where ℓ is limited to D_k).

The interference $I_k(r_k^q, r_k^q + \ell)$ on a copied job $J_k^{q,p}$ in the interval $[r_k^q, r_k^q + \ell)$ is defined as the cumulative length of all the intervals in which $J_k^{q,p}$ is ready to be executed but cannot be scheduled on any processor owing to m higher priority jobs [12].

When interference occurs on $J_k^{q,p}$ at a certain time instance, at least m higher priority jobs exist to hinder the execution of $J_k^{q,p}$. Thus, we need to calculate how much of the execution of the higher priority job contributes to the interference $I_k(r_k^q, r_k^q + \ell)$ on J_k^q to upper bound $I_k(r_k^q, r_k^q + \ell)$. An important property of schedules under TL-NMR is that the execution of a copied job $J_k^{q,p}$ can be hindered by jobs of not only other tasks τ_i but also by other copied jobs (of τ_k) sharing the same release time r_k^q and absolute deadline d_k^q. For example, $J_3^{1,2}$ in Figure 1b cannot execute in the interval [2, 4) owing

to $J_2^{1,1}$, $J_2^{1,2}$, (of τ_2) and $J_3^{1,1}$ (of τ_3) occupying three processors. Hence, we first let J_i^p be the set of the p-th copied jobs of τ_i. Then, we define the interference $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ of J_i^p on $J_k^{q,p}$ and the interference $I_{k\leftarrow k}(r_k^q, r_k^q+\ell)$ of the other copied job $J_k^{q,g}$ (i.e., except the job of interest $J_k^{q,p}$) of τ_k on $J_k^{q,p}$ in an interval $[r_k^q, r_k^q+\ell)$.

The interference $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ of J_i^p on $J_k^{q,p}$ in the interval $[r_k^q, r_k^q+\ell)$ is defined as the cumulative length of all the intervals in which $J_k^{q,p}$ is ready to be executed but cannot be scheduled on any processor while jobs of J_i^p execute.

The interference $I_{k\leftarrow k}(r_k^q, r_k^q+\ell)$ of the other copied job $J_k^{q,g}$ (i.e., except the job of interest $J_k^{q,p}$) of τ_k on $J_k^{q,p}$ in the interval $[r_k^q, r_k^q+\ell)$ is defined as the cumulative length of all the intervals in which $J_k^{q,p}$ is ready to be executed but cannot be scheduled on any processor while $J_k^{q,g}$ executes.

To upper bound $I_{k\leftarrow i}(r_k^q, r_k^q+\ell)$ of J_i^p on $J_k^{q,p}$, RTA for TL-NMR uses the notion of the workload of J_i^p in an interval of length ℓ, which is defined as the amount of computation time required for J_i^p in ℓ [13]. The upper part of Figure 2 illustrates the scenario in which the workload of a task τ_i is maximized under preemptive work-conversing TL-NMR scheduling. Let us assume that $p = 1$. The first job $J_i^{q,1}$ of τ_i, in the upper side of Figure 2, starts its execution at the beginning of the interval ℓ and completes its execution at d_i^q, thereby executing for C_i time units without any interference or delay. Thereafter, the following jobs (i.e., $J_i^{q+1,1}$ and $J_i^{q+2,1}$) are released and scheduled subsequently. Considering the number of executions of jobs fully executing for C_i, the other jobs executing for a portion of C_i, and the number of copies N_i, the upper-bounded workload $W_i(\ell)$ is calculated by

$$I_{k\leftarrow i}(r_k^q, r_k^q+\ell) = W_i(\ell) = F_i(\ell)\cdot C_i + min\Big(C_i, \ell + D_i - C_i - F_i(\ell)\cdot T_i\Big), \tag{4}$$

where $F_i(\ell)$ is the number of jobs executing for C_i calculated by

$$F_i(\ell) = \left\lfloor \frac{\ell + D_i - C_i}{T_i} \right\rfloor. \tag{5}$$

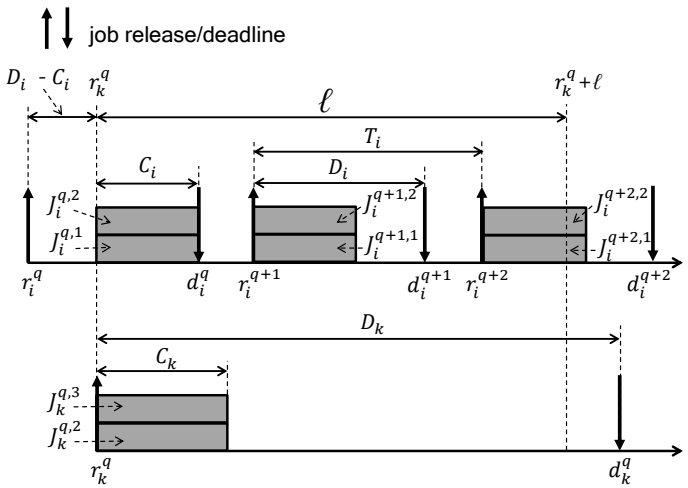


Figure 2. Worst case scenario in which workload of J_i^p (set of p-th copied jobs of τ_k) and execution of $J_k^{q,g}$ (the other copied job $J_k^{q,g}$ of τ_k except $J_k^{q,p}$) are maximized in an interval $[r_k^q, r_k^q+\ell)$ under preemptive work-conserving TL-NMR scheduling for $N_i = 2$ and $N_k = 3$.

Because the other copied jobs of τ_k share the same release time r_k^q and absolute deadline d_k^q, ℓ is limited to D_k. The interference $I_{k\leftarrow k}(r_k^q, r_k^q+\ell)$ of the other copied job $J_k^{q,g}$ of τ_k on $J_k^{q,p}$ in the interval $[r_k^q, r_k^q+\ell)$ can be upper-bounded by

$$I_{k\leftarrow k}(r_k^q, r_k^q+\ell) = min(C_k, \ell), \tag{6}$$

as shown in the lower part of Figure 2. As seen in Figure 2, N_i times of J_i^p and $N_k - 1$ times of $J_k^{q,g}$ can contribute to $I_k(r_k^q, r_k^q + \ell)$. In addition, because a job cannot execute in a time instant if m other higher priority jobs execute, using upper-bounded $I_{k \leftarrow i}(r_k^q, r_k^q + \ell)$ and $I_{k \leftarrow k}(r_k^q, r_k^q + \ell)$, $I_k(r_k^q, r_k^q + \ell)$ is upper-bounded by

$$I_k(r_k^q, r_k^q + \ell) = \left\lfloor \frac{1}{m} \cdot \left(\sum_{\tau_i \in \tau - \{\tau_k\}} N_i \cdot \left(I_k^i(r_k^q, r_k^q + \ell) \right) + (N_k - 1) \cdot I_{k \leftarrow k}(r_k^q, r_k^q + \ell) \right) \right\rfloor. \tag{7}$$

Therefore, if $I_k(r_k^q, r_k^q + \ell)$ is not larger than $\ell - C_k$, $J_k^{q,p}$ can finish its execution at or before $r_k^{q,p} + \ell$, where ℓ is limited to D_k.

Although (7) safely upper-bounds $I_k(r_k^q, r_k^q + \ell)$, the value derived by (7) is highly overestimated because it includes the amount of execution of J_i^p and $J_k^{q,g}$, which can be performed in parallel with $J_k^{q,p}$. As seen in Figure 3, a portion of the execution of J_i^p is performed in parallel with a job $J_k^{q,1}$ of interest, which cannot contribute to $I_k(r_k^q, r_k^q + \ell)$. This phenomenon will also occur with $J_k^{q,2}$ or $J_k^{q,3}$ if C_k is larger than $\ell - C_k$. Thus, RTA for TL-NMR limits the amount of $I_{k \leftarrow i}(r_k^q, r_k^q + \ell)$ of J_i^p (similarly $I_{k \leftarrow k}(r_k^q, r_k^q + \ell)$ of $J_k^{q,g}$), which potentially contributes to $I_k(r_k^q, r_k^q + \ell$ and $\ell - C_k + 1$. Here, $I_k(r_k^q, r_k^q + \ell$ is more tightly upper-bounded than (7) and is given by

$$\begin{aligned} I_k(r_k^q, r_k^q + \ell) = \Bigg\lfloor \frac{1}{m} \cdot \Bigg(& \sum_{\tau_i \in \tau - \{\tau_k\}} N_i \cdot \left(min\left(I_k^i(r_k^q, r_k^q + \ell), \ell - C_k + 1 \right) \right) \\ & + (N_k - 1) \cdot min\left(I_{k \leftarrow k}(r_k^q, r_k^q + \ell), \ell - C_k + 1 \right) \Bigg) \Bigg\rfloor. \end{aligned} \tag{8}$$

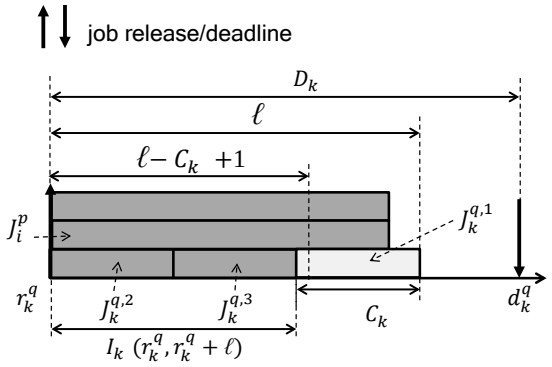


Figure 3. Scenario in which the amount of interference $I_{k \leftarrow i}(r_k^q, r_k^q + \ell)$ of J_i^p that can contribute to $I_k(r_k^q, r_k^q + \ell)$ on job $J_k^{q,p}$ is limited to $\ell - C_k + 1$ for a task set $\tau = \{\tau_i, \tau_k\}$, and $N_i = 2$ and $N_k = 3$.

Based on this reasoning, RTA for TL-NMR tests the schedulability of τ_k as follows.

Theorem 1. *A task $\tau_k \in \tau$ is schedulable under TL-NMR, if a copied job $J_k^{q,p}$ satisfies the following for any ℓ that holds $C_k \leq \ell \leq D_k$.*

$$C_k + I_k(r_k^q, r_k^q + \ell) \leq \ell. \tag{9}$$

Proof. Suppose that $J_k^{q,p}$ cannot complete its execution in $[r_k^q, r_k^q + \ell)$ even if Equation (9) holds. By the definition of $I_k(r_k^q, r_k^q + \ell)$, $J_k^{q,p}$'s execution is hindered by higher priority jobs in $[r_k^q, r_k^q + \ell)$ for at most $I_k(r_k^q, r_k^q + \ell)$. In addition, the worst case execution time for $J_k^{q,p}$ to complete its execution is C_k. Thus, $J_k^{q,p}$ can complete its execution in $C_k + I_k(r_k^q, r_k^q + \ell)$ after the release of r_k^q. By Equation (9), $C_k + I_k(r_k^q, r_k^q + \ell) \leq \ell$ holds, and ℓ is less than or equal to D_k. Therefore $J_k^{q,p}$ is schedulable if Equation (9) holds, which contradicts the supposition. □

The remaining issue is to find a value of ℓ and an upper bound $I_k^i(r_k^j, r_k^j + \ell)$. RTA for TL-NMR works as follows. Initially, ℓ is set to C_k and RTA tests whether the inequality holds. If the inequality holds, the task is deemed schedulable. Otherwise, RTA resets ℓ to the previous value of the LHS of the inequality, until the inequality holds or $\ell > D_k$; $\ell > D_k$ represents that τ_k is deemed unschedulable. If the inequality holds, τ_k is deemed schedulable and the value of ℓ satisfying the inequality is R_k, i.e., $R_k \leq D_k$ holds.

Figure 4 illustrates how RTA judges the schedulability of τ_k scheduled by TL-NMR.

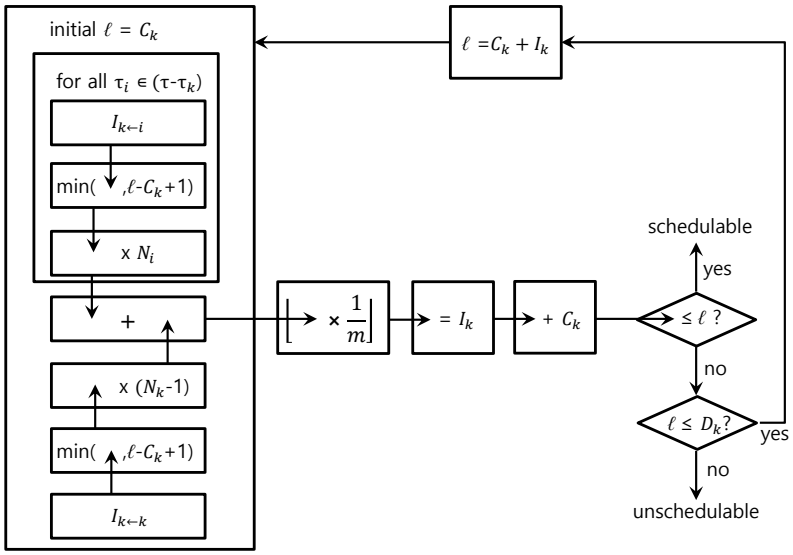


Figure 4. Flux diagram illustrating how RTA judges the schedulability of τ_k scheduled by TL-NMR.

5. Results

In this section, we evaluate the performance of the proposed TL-NMR compared to that of the existing fault-tolerant techniques. For performance metrics, we measure the number of randomly generated task sets that are deemed schedulable (called schedulable ratio) and the average of considered task sets' system safety (called average system safety). Task sets for our evaluation are randomly generated based on a popular task set generation method that has been exploited in a number of existing studies regarding real-time scheduling [14–16]. The task set generation method used in our study has two parameters such that the number of processors $m \in \{2, 4, 8, 16\}$, and the input parameter α or $1/\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for individual task use (C_i/T_i) distribution. If α is given, a value for C_i/T_i is uniformly selected in [0, 0.5) and [0.5, 1) with probability α and $1 - \alpha$, respectively, according to a given bimodal distribution. On the other hand, when $1/\beta$ is given, the value is selected according to the exponential distribution whose probability density function is $\beta \cdot exp(-\beta \cdot x)$. Then, the parameters of a task in a task set are determined as follows. T_i is uniformly determined in $[1, 1000]$, C_i is chosen by the bimodal or exponential parameter with T_i already determined, and D_i is uniformly chosen in $[C_i, T_i]$. Ten thousand task sets are randomly generated for each value of m.

Next, we discuss the performance (i.e., schedulable ratio and average system safety) and properties of the following techniques:

- TL-NMR-RM: RM scheduling incorporating the proposed TL-NMR is discussed in Section 3, and
- xMR-RM: RM scheduling incorporating NMR in which x identical copies are executed to improve reliability (e.g., 3M-RM represents the scheduler where three identical copies of each job are executed under RM scheduling) proposed in the existing studies [4–6].

Please note that although we conducted experiments for well performing FP scheduling such as earliest quasi-deadline first and deadline monotonic, we do not explicitly discuss the performance of such scheduling because they show a trend similar to RM scheduling. 1MR-RM may also represent

RM scheduling without NMR because a single copy of each job of each task is executed under vanilla RM scheduling.

Figure 5 presents the experimental results regarding the schedulable ratio and average system safety of the considered techniques. Figure 5a,b shows the schedulable ratio of the considered techniques according to varying task set use ($\sum_{\tau_k \in \tau} C_k/T_k$) for $m = 2$ and $m = 16$, respectively. TOT represents the number of generated task sets whose task set use is represented by the axis of task set use. For example, the number of generated task sets whose task set use is about 1.5 is approximately 430 (represented by z-axis). Furthermore, Figure 5c–f plots the average system safety of task sets of the considered techniques for $\gamma = 0.001$ and $\gamma = 0.01$, respectively.

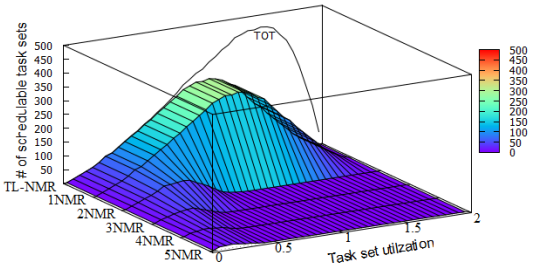
From the figures, we make the following observations:

- TL-NMR-RM maintains the same performance in schedulable ratio (i.e., the number of task sets deemed schedulable) compared to 1MR-RM for both $m = 2$ and 16 (Figure 5a,b),
- TL-NMR-RM improves the average system safety more for all task set use compared to 1MR-RM, while higher number of job-copies in NMR decreases both the schedulable ratio and average system safety (Figure 5c–f), and
- TL-NMR-RM better outperforms 1MR-RM for greater values of γ (Figure 5c–f).

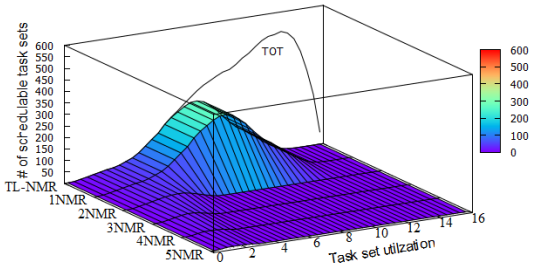
Observation 1 is due to the key property of TL-NMR. That is, it improves the system reliability without schedulability loss by determining the individual number of copies of each task (in conjunction with the proposed RTA in Section 4) according to Algorithm 1. Thus, a schedulable task set under 1MR-RM (i.e., vanilla RM) never become unschedulable under TL-NMR-RM.

Observation 2 is the natural consequence stemming from Observation 1 in that the system reliability is enhanced owing to increased N_k (as Equation (2) indicates), while the schedulability is not changed under TL-NMR-RM compared to that in 1MR-RM. However, the average system safety decreases for the higher number of job-copies in NMR. This counterintuitive phenomenon happens because the schedulability plays a more important role to improve the average system safety than to increase the system reliability. Recall that the system safety is zero when the task set is unschedulable. System reliability is inevitably improved if N_k of each task increases, but the schedulability is not guaranteed when such an increase of N_k is conducted not in conjunction with schedulability analysis such as RTA. Thus, the higher number of copies in NMR aggravates the average system safety by compromising schedulability even though it may improve reliability.

Observation 3 highlights the advantage of TL-NMR such that the system reliability under TL-NMR increases more compared to that under 1MR-RM when the arrival rate of transient error increases. As Equation (2) indicates, the system reliability is disproportional to γ but proportional to N_k. Thus, the average system safety under 1MR-RM (and the other xMR-RM series) sharply decreases with increasing γ, while TL-NMR-RM makes up for such a degradation by increasing N_k without schedulability loss.

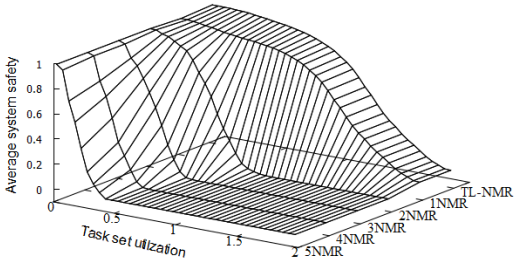


(**a**) for $m = 2$

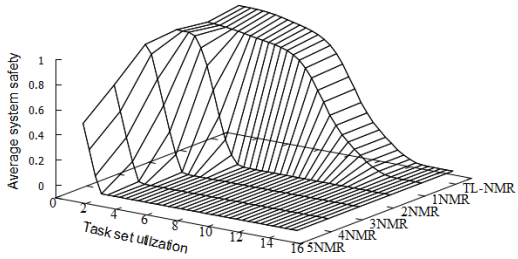


(**b**) for $m = 16$

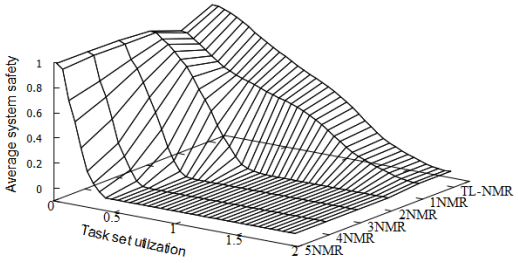
Figure 5. *Cont.*



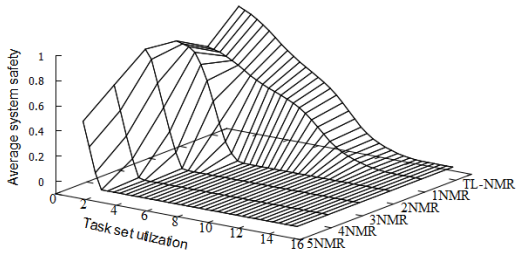
(**a**) for $m = 2$ and $\gamma = 0.001$



(**b**) for $m = 16$ and $\gamma = 0.001$



(**c**) for $m = 2$ and $\gamma = 0.01$



(**d**) for $m = 16$ and $\gamma = 0.01$

Figure 5. Evaluation results of considered *-RM series for $m = 2$ and 16.

6. Conclusions

In this study, we proposed a TL-NMR technique that improves the system reliability of a target system in which tasks are scheduled by any FP scheduling without schedulability loss. TL-NMR overcomes a limitation of the existing NMR, i.e., some tasks may miss their deadlines while multiple copies of tasks are executed in parallel to improve system reliability. The TL-NMR framework determines the number of copies of each job J_k^q of a task τ_k while ensuring that every copied job completes its execution before its absolute deadline in conjunction with an RTA framework proposed in this paper. Through our experiments, we demonstrated that TL-NMR maintains schedulability while it improves average system safety compared to the existing vanilla RM scheduling technique. The performance gap between TL-NMR and vanilla RM becomes larger for systems where transient error occurs frequently (i.e., for a larger value of γ). For future work, we plan to extend our work to consider multiple shared resources and derive resource-locking protocols [17,18] to improve the analytical capability.

Author Contributions: Conceptualization, J.B. (Jaemin Baek), J.B. (Jeonghyun Baek) and H.B.; Software, H.B.; data curation, J.B. (Jaemin Baek) and J.B. (Jeonghyun Baek); writing-original draft preparation, J.B. (Jaemin Baek), J.B. (Jeonghyun Baek), J.Y. and H.B.; writing-review and editing, J.Y.; Supervision, H.B.; project administration, H.B.; funding acquisition, H.B.

Funding: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2019R1F1A1059663).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, C.; Layland, J. Scheduling Algorithms for Multi-programming in A Hard-Real-Time Environment. *J. ACM* **1973**, *20*, 46–61. [CrossRef]
2. Ekpo, S.; George, D. A system-based design methodology and architecture for highly adaptive small satellites. In Proceedings of the IEEE International Systems Conference, San Diego, CA, USA, 5–8 April 2010; pp. 516–519.

3. Randell, B. System structure for software fault tolerance. *IEEE Trans. Softw. Eng.* **1975**, *SE-1*, 220–232. [CrossRef]
4. Malhotra, S.; Narkhede, P.; Shah, K.; Makaraju, S.; Shanmugasundaram, M. A review of fault tolerant scheduling in multicore systems. *Int. J. Sci. Technol. Res.* **2015**, *4*, 132–136.
5. Yu, X.B.; Zhao, J.S.; Zheng, C.W.; Hu, X.H. A Fault-Tolerant Scheduling Algorithm using Hybrid Overloading Technology for Dynamic Grouping based Multiprocessor Systems. *Int. J. Comput. Commun. Control.* **2012**, *7*, 990–999. [CrossRef]
6. Zhou, J.; Yin, M.; Li, Z.; Cao, K.; Yan, J.; Wei, T.; Chen, M. Fault-Tolerant Task Scheduling for Mixed-Criticality Real-Time Systems. *J. Circuits Syst. Comput.* **2017**, *26*, 1–17. [CrossRef]
7. Ghosh, S.; Melhem, R.; Mosse, D. Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems. *IEEE Trans. Parallel Distrib. Syst.* **1997**, *8*, 272–284. [CrossRef]
8. Manimaran, G.; Murthy, C.S.R. A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis. *IEEE Trans. Parallel Distrib. Syst.* **1998**, *9*, 1137–1152. [CrossRef]
9. Al-Omari, R.; Somani, A.K.; Manimaran, G. Efficient overloading techniques for primary-backup scheduling in real-time systems. *J. Parallel Distrib. Comput.* **2004**, *64*, 629–648. [CrossRef]
10. Cirinei, M.; Bini, E.; Lipari, G.; Ferrari, A. A Flexible Scheme for Scheduling Fault-Tolerant Real-Time Tasks on Multiprocessors. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium, Rome, Italy, 26–30 March 2007; pp. 1–8.
11. Aminzadeh, S.; Ejlali, A. A comparative study of system-level energy management methods for fault-tolerant hard real-time systems. *IEEE Trans. Comput.* **2011**, *60*, 1228–1299. [CrossRef]
12. Bertogna, M.; Cirinei, M.; Lipari, G. Improved Schedulability Analysis of EDF on Multiprocessor Platforms. In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), Balearic Islands, Spain, 6–8 July 2005; pp. 209–218.
13. Bertogna, M.; Cirinei, M. Response-Time Analysis for globally scheduled Symmetric Multiprocessor Platforms. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Tucson, AZ, USA, 3–6 December 2007.
14. Bertogna, M.; Cirinei, M.; Lipari, G. Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms. *IEEE Trans. Parallel Distrib. Syst.* **2009**, *20*, 553–566. [CrossRef]
15. Baker, T.P. *Comparison of Empirical Success Rates of Global vs. Partitioned Fixed-Priority EDF Scheduling for Hard Real-Time*; Technical Report TR–050601; Department of Computer Science, Florida State University: Tallahassee, FL, USA, 2005.
16. Andersson, B.; Bletsas, K.; Baruah, S. Scheduling Arbitrary-Deadline Sporadic Task Systems on Multiprocessor. In Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Barcelona, Spain, 30 November–3 December 2008; pp. 197–206.
17. Sha, L.; Rajkumar, R.; Lehoczky, J.P. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Trans. Comput.* **1990**, *39*, 1175–1185. [CrossRef]
18. Easwaran, A.; Adersson, B. Resource Sharing in Global Fixed-Priority Preemptive Multiprocessor Scheduling. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Washington, DC, USA, 1–4 December 2009; pp. 377–386.