

1997-12-01

18th IEEE Real-Time Systems Symposium: Work in Progress Sessions

Bestavros, Azer. "Proceedings of the 18th Real-Time Systems Symposium WIP Session",
Technical Report BUCS-1997-021, Computer Science Department, Boston University,
December 1, 1997. [Available from: <http://hdl.handle.net/2144/3748>]

<https://hdl.handle.net/2144/3748>

"Downloaded from OpenBU. Boston University's institutional repository."

Predictability and Responsiveness of Fault Recovery Operations in Real-Time Systems

Pedro Mejía-Alvarez
CINVESTAV-IPN. Sección de Computación
Av. I.P.N. 2508, Zacatenco.
México, DF. 07300
pmejia@computacion.cs.cinvestav.mx

Juan A. De La Puente
E.T.S.I.Telecomunicación
Universidad Politécnica de Madrid
Madrid, España. E28040
jpuente@dit.upm.es

1 Introduction

This paper focuses on the problem of scheduling real-time and transient fault tolerant requests in a uniprocessor environment, using time redundancy. The problem of scheduling recovery time and replicate resources in a real-time system to enhance its fault tolerance, requires developing efficient scheduling policies, as well as fault tolerant mechanisms that resolve resource contention conflicts among different tasks requesting access to a shared resource such as the processor [2]. This is a complex problem because these scheduling policies and fault tolerant mechanisms must allow the scheduler *to meet two competing objectives*: ensure that the timing requirements of the real-time application workload are met, and ensure that the timing requirements of any recovery operation triggered by the detection of errors are met. While the scheduler strives to meet both of these objectives, there may be situations in which it may be unable to do so, these situations may be, (a) overload or high utilization of the real-time workload, (b) high rate of faults, and (c) unpredictability of fault recovery operations. An appropriate solutions must handle these conditions. Sources of unpredictability and bounds of execution must be identified and solved. Predictable mechanisms for fault tolerance must be developed, and schedulability analysis must be designed for integrating time-critical constraints of real-time workloads, and responsiveness-based priorities to recovery operations. All scheduling decisions must be performed considering that the timing correctness of the most critical real-time tasks is maintained and the state of the system is always predictable, even if some of the tasks are forced to suffer performance degradation. This paper describes ways to characterize the recovery workload providing the solutions mentioned above, and developing schedulability analysis for evaluating the timing correctness of a variety of techniques for fault tolerance. Also, a criterion for responsiveness of fault recovery operations is developed, motivated by the need to guarantee a graceful degradation on the real-time workload during recovery.

2 Schedulability Analysis

We are considering a Rate Monotonic Scheduling model[4], with sets of periodic preemptive tasks with fixed priorities. Tasks are independent and have no precedence constraints. We assume that faults are transient and only one task is being affected by each fault. In our fault model, one fault can be tolerated by any task if it occurs within a minimum inter-arrival time T^F . Therefore, faults can be tolerated by Recovery Blocks (RB)[4], Exception Handling (EH)[4], Checkpointing[4], Imprecise Computations (IC)[3] and Checkpointed Imprecise Computations (Ch-IC). Schedulability is the first step in the process of designing a system capable of predicting and guaranteeing real-time constraints. Lehoczky [5] formulated a necessary and sufficient schedulability test for fixed priority real-time task sets under critical instant phasing. This test dictates that task τ_i is schedulable if,

$$\min_{\{0 < t < D_i\}} \{W_i(t)/t\} \leq 1 \quad (1)$$

where $W_i(t) = \sum_{j=1}^i C_j \cdot \lceil t/T_j \rceil$. $W_i(t)$ denotes the cumulative workload that has arrived from priority levels 1 to i in the time interval $[0, t]$ under critical instant phasing. It follows that the entire task set is schedulable if,

$$\max_{\{1 \leq i \leq n\}} \min_{\{t \in S_i\}} \{W_i(t)/t\} \leq 1 \quad (2)$$

S_i is referred to as the set of scheduling points [5].

Now, we will extend this framework to evaluate the timing impact caused to a real-time workload when different fault tolerant mechanisms are supported. The addition of fault tolerant constraints in the analysis implies an increase in the cumulative workload. The fault tolerant cumulative workload, $W_i^F(t)$, is given by,

$$W_i^F(t) = \left(\sum_{j=1}^i \lceil t/T_j \rceil (C_j + O_j) \right) + C_i^F \cdot \lceil t/T^F \rceil \quad (3)$$

C_i^F denotes the recovery time reserved for task τ_i , T^F denotes a fixed fault inter-arrival time for the task set, and O_j denotes the increase in execution time of task τ_j

because of a checkpoint. In the remainder of this section, an schedulability analysis is developed for the following fault tolerant mechanisms:

- **Recovery Blocks:** $C_i^F = \max_{\{1 \leq k \leq i\}} C_k^S$.

C_k^S is the computation time of the secondary block which considers the occurrence of only one fault during the execution of the primary block of task τ_k or during the execution of the primary block of some higher priority task which has preempted τ_k . We are assuming that each task has only one secondary block. From equation (3), in this case, C_i^F denotes the computation time of the primary block.

- **Forward Recovery:** $C_i^F = \max_{\{1 \leq k \leq i\}} C_k^F$.

In this model, C_k^F is the computation time of the forward recovery mechanism of task τ_k .

- **Checkpointing:** $C_i^F = \max_{\{1 \leq k \leq i\}} L_k$.

In this model, every task includes an additional processing time due to n_i checkpoints of size o_i . Therefore, the execution time of τ_i increases $O_i = (n_i \times o_i)$. Recovery is performed by rolling back to the most recent checkpoint. If L_i denotes the longest checkpoint interval of task τ_i , then recovery of task τ_i (if a fault occurs in task τ_i or in some higher priority task which has preempted it) will add $\max_{\{1 \leq k \leq i\}} L_k$ to the response time of τ_i .

- **IC:** $C_i^F = \max_{\{1 \leq k \leq i\}} [0, (m_k - p_k)]$.

In the imprecise computation model, task τ_i has an execution time C_i , and it consists in a mandatory part of length m_i and an optional part p_i of length $p_i = C_i - m_i$. An acceptable level of accuracy is reached after executing m_i units of time in task τ_i . During the optional part, the result is improved until a precise result is reached after p_i units of execution. We assume that whenever time is not available to execute task τ_i to completion, either the entire optional part or portions of it can be skipped, without affecting the correctness of the system. If a fault occurs within the mandatory part of task τ_i (or in some higher priority task which has preempted τ_i), only the mandatory part will be re-executed. Therefore, recovery will add $\max_{\{1 \leq k \leq i\}} [0, (m_k - p_k)]$ to the response time of τ_i .

- **Ch-IC:** $C_i^F = \max_{\{1 \leq k \leq i\}} [0, (Lm_k - p_k)]$

The checkpointed imprecise computation model includes checkpoints only in the mandatory part. The analysis is similar to the Imprecise computation model. Lm_i denotes the longest checkpoint interval within the mandatory part of task τ_i . If a fault occurs within the mandatory part of task τ_i (or in some higher priority task which has preempted τ_i),

only the mandatory part will be re-executed. Therefore, recovery will add $\max_{\{1 \leq k \leq i\}} [0, (Lm_k - p_k)]$ to the response time of τ_i .

As stated in equation (2), we can compute the schedulability test for the fault tolerant real-time task set as,

$$\max_{\{1 \leq i \leq n\}} \min_{\{l \in S_i\}} \{W_i^F(t)/t\} \leq 1 \quad (4)$$

Equation (4) gives us an schedulability condition for fault tolerant real-time task sets.

Table 1 shows a task set consisting of 3 periodic tasks, and its associated timing characteristics for each fault tolerant mechanism. From this requirements, in Table 2 we show the schedulability analysis for the fault tolerant models previously discussed. Here, we have assumed a fault inter-arrival time of $T^F = 350$. It is possible to observe in Table 2 that under these models, only task τ_3 misses its deadline under the checkpointing model.

	Fault-Free Workload		Recovery Workload						
	C_i	T_i	BB C_i^F	F-H C_i^F	Checkpoints $C_i + O_i$	L_i	IC m_i	p_i	Ch-IC Lm_i
τ_1	7	20	5	2	8	4	4	3	2
τ_2	10	40	8	3	11	6	6	4	3
τ_3	20	80	11	2	21	11	8	12	4

Table 1: Timing Characteristics of the Real-Time Workload and its Recovery

2.1 Bounds for Recovery Operations

Another step towards developing a predictable system considers bounding the effect of the recovery workload. We have developed two bounds for recovery operations. The first bound, U_f is an extension of Liu and Layland utilization bound [4] and [1]. The second bound is the breakdown utilization bound of a tasks set, U_{BD} .

Liu and Layland [4] proved that if tasks have fixed priorities that are derived from their periods, then any set of n tasks with a total utilization U_W below $n(2^{1/n} - 1)$ is schedulable in a uniprocessor system. Recently, Ghosh[1] developed an utilization bound for fault tolerant task sets, where recovery is done by re-execution. We are extending his work to include recovery bounds that have been obtained from the various fault tolerant schemes previously discussed. In our approach, we have included the fault recovery utilization factor U_B into the fault tolerant recovery bound, U_f , which is given by,

$$U_f = (n-1)(1-U_B)([2(1-U_B)]^{1/n-1} - 1) + U_B \quad (5)$$

where $U_B = C^F/T^F$. C^F denotes the maximum recovery time of the task set, and T^F denotes the fault inter-arrival time, and its value is $C^F = \max_{j=1}^n C_j^F$. We assume that $T^F = 350$. If $U_W \leq U_f$ then the task set is schedulable. Derivation of U_f is provided by [1].

The breakdown utilization of a task set U_{BD} , is the utilization in which real-time application fully utilizes the processor [4]. Lehoczky introduced an expression

Task	Fault-free	RB	Excep.	Checkp	IC	Ch-IC
τ_1	0.35	0.6	0.45	0.6	0.4	0.4
τ_2	0.6	0.8	0.675	0.825	0.65	0.675
τ_3	0.85	0.987	0.887	1.075	0.875	0.938
U_B	0	0.03	0.009	0.03	0.005	0
U_W	0.85	0.85	0.85	0.937	0.85	0.937
U_f	0.78	0.792	0.817	0.792	0.822	0.779
U_{BD}	1.0	0.861	0.958	0.871	0.97	0.998

Table 2: Analysis under faulty conditions: $\min_{t \in S_i} \{W_i(t)/t\}$ and Utilization Bounds for Recovery

for computing the breakdown utilization of a task set [5]. The breakdown utilization U_{BD} of a task set with utilization $U_W = \sum_{i=1}^n C_i/T_i$, including fault-induced requirements is given by

$$U_{BD} = \frac{U_W}{\max_{1 \leq i \leq n} \min_{t \in S_i} \{W_i^F(t)/t\}} \quad (6)$$

If $U_W \leq U_{BD}$ then the task set is schedulable, otherwise the task set is unschedulable.

Table 2 shows the fault tolerant recovery bound U_f , and the breakdown utilization U_{BD} , for the task set used in Table 1. It can be observed that under all the recovery schemes $U_W > U_f$. On the other hand, if we compare U_{BD} with U_W , only in the checkpointing model, the task set is unschedulable.

2.2 Computing the largest recovery time

In this section we are interested in calculating: *what is the largest recovery time LR_i for tasks τ_i with period T_i^F that can be added and still guarantee the schedulability of the workload [2].* From equation (1), S_i^* is defined as the subset of scheduling points t for τ_i which will allow it to tolerate the addition of recovery time under critical instant phasing,

$$S_i^* = \{t \in S_i \mid W_i(t)/t < 1\} \quad (7)$$

The additional recovery time, $\Delta W_i(t)$, tolerated at each scheduling point is computed by,

$$\Delta W_i(t) \mid_{t \in S_i^*} = \{t - W_i(t)\} \quad (8)$$

Any of the elements S_i^* guarantees schedulability as long as the recovery time accumulated from higher priority tasks during $[0, t]$ does not exceed $\Delta W_i(t)$. The largest recovery time with priority i which arrives $\lceil t/T_i^F \rceil$ in the interval $[0, t]$ may be calculated by,

$$LR_i = \max_{t \in S_i^*} \{\Delta W_i(t) / \lceil t/T_i^F \rceil\} \quad (9)$$

It follows that the largest recovery time which may be added to a set of n tasks without disturbing their schedulability is given by,

$$LT_i = \min_{1 \leq k \leq n} \max_{t \in S_k^*} \{\Delta W_k(t) / \lceil t/T_i^F \rceil\} \quad (10)$$

Example 1. Considering the fault-free workload presented in Table 1, the analysis for calculating the largest recovery time is given in Table 3. Table 3 shows that there is enough spare time in the schedule to satisfy the timing requirements of the fault tolerant recovery workload, except for the checkpointing model. Using the computation time for the checkpointing model (see Table 1), we obtained $LR_1 = 12$, $LR_2 = 13$, and $LR_3 = 5$. From this results, it can be seen that task τ_3 misses its deadline due to the fact that the recovery time L_3 is greater than LR_3 .

t	$W_1(t)$	$\Delta W_1(t)$	$W_2(t)$	$\Delta W_2(t)$	W_3	$\Delta W_3(t)$
20	7	13	17	3	37	
40			24	16	44	
60					61	
80					68	12
LR_i		13.0		16.0		12.0

Table 3: Largest Recovery Time Analysis

3 Responsiveness of Fault Recovery

Responsiveness of fault recovery operations, is a property of the scheduler which guarantees in a faulty situation, that the most critical tasks remain schedulable even if the deadlines of some tasks cannot be met. To solve this problem, the scheduler must be capable of extracting information in the schedule about the response time of recovery operations, when a fault occurs.

Response time of faulty recovery operations is based upon the priority in which faulty tasks are serviced. Faulty tasks serviced at lower priorities will have longer response times and a greater chance of missing its deadlines. On the other hand, faulty tasks will have shorter response times if they are serviced at higher priorities. To anticipate fault situations in real-time tasks and increase their responsiveness, it is necessary that the scheduler have enough information about their criticalness. Therefore, the decision of servicing recovery operations at lower or higher priorities must be based on criticalness of real-time tasks. This information will allow the scheduler to control load shedding decisions during transient recovery overloads.

Based on the above observations, we have developed an scheduling tool for assisting the scheduler during recovery, at different levels of responsiveness. The RTAB Table contains timing information about different responsiveness levels for servicing recovery operations. For a given real-time and recovery workload, generating the RTAB table is a two step process, involving the computation of the different levels of responsiveness, and its integration into the RTAB Table. The next sections present an analysis for calculating different levels of responsiveness, and an example of the integration of different levels of responsiveness into the RTAB Table.

3.1 Computing Levels of Responsiveness

We have classified responsiveness of fault recovery according to the following criterion.

If a fault occurs at time t_F within the execution of task τ_i , the recovery operation (with computation time C_i^F) must be serviced before d_{ij} (deadline of the j^{th} activation of task τ_j). Therefore, at time t_F the scheduler must find slack time SL_i within $[t_F, d_{ij}]$ for the recovery operation. To service the recovery operation, we have developed the following different levels of responsiveness,

Fair, indicates that recovery is scheduled at the lowest possible priority, such that any tasks miss its deadlines.

$$FA_i(t_F) = \begin{cases} \min_{j=i}^n \{SL_j(t_F)\} & \text{if } \min_{j=i}^n \{SL_j\} \geq C_i^F \\ 0 & \text{otherwise} \end{cases}$$

Greedy early, indicates that recovery is scheduled at a higher priorities without making any task miss its deadline.

$$GE_i(t_F) = \begin{cases} \min_{j=1}^n \{SL_j(t_F)\} & \text{if } \min_{j=1}^n \{SL_j\} \geq C_i^F \\ 0 & \text{otherwise} \end{cases}$$

Gracefully late, indicates that scheduling recovery operations is only possible at lower priorities, making lower priority tasks miss their deadlines.

$$GL_i(t_F) = \begin{cases} \min_{j=1}^n \{SL_j(t_F)\} & \text{if } \min_{j=1}^n \{SL_j\} \geq C_i^F \\ 0 & \text{otherwise} \end{cases}$$

Critically late, indicates that scheduling recovery operations is only possible at higher priorities. This could make higher priority tasks miss their deadlines.

$$CL_i(t_F) = \begin{cases} \min_{j=1}^n \{SL_j(t_F)\} & \text{if } \min_{j=1}^n \{SL_j\} \geq C_i^F \\ (d_{ij} - t_F) & \text{else if } (d_{ij} - t_F) \geq C_i^F \\ 0 & \text{otherwise} \end{cases}$$

Too late recovery, indicates that recovery is not possible, even scheduling the recovery operation at the highest priority. This level is obtained when $CL_i(t_F) = 0$.

SL_i denotes the *Slack* of task τ_i , which is computed by,

$$SL_i(t_F) = d_{ij} - t_F - CW_i(t_F, d_{ij}) \quad (11)$$

where CW_i denotes the cumulative workload in $[t_a, t_b]$,

$$CW_i(t_a, t_b) = \sum_{k=1}^n C_k \times ([t_b/T_k] - [t_a/T_k]) - c_{k,\tau_a} - c_{k,\tau_b} - tr_{k,t_a} \quad (12)$$

where c_{k,t_a} is the computation time of the completed portion of task τ_k at time t_a . c_{k,τ_b} is the computation time of the remaining portion of task τ_k at time t_b . It can be determined by constructing the schedule of periodic tasks for an hyperperiod. tr_{k,t_a} is the computation time of the remaining portion of task τ_k at time t_a .

3.2 The RTAB Responsiveness Table

To assist the scheduler in performing on-line scheduling decisions, we have developed a responsiveness table containing information about the responsiveness attainable during recovery. The RTAB Table is organized according to the responsiveness classification described in the previous section. To describe the RTAB Table, we present the following example.

Example 3: Consider the case in which task τ_2 fails at time 12, causing a recovery request of size $C_i^F = 11$ (for the Recovery Block mechanism) to be issued to the scheduler. This gives us the following timing situation,

$$t_F = 12, t_b = d_{2,1} = 40, \text{ and } C_i^F = 11. \\ S_1(12) = 31, S_2(12) = 31, S_3(12) = 17.$$

It is possible to observe in Table 4 that if *Recovery Blocks* are used to handle faults, due to the fact that the computation time of the secondary block of task τ_2 is equal to $C_2^s = 11$ (see Table 1), it can recover under any level of responsiveness.

CL	GL	GE	FA
31	31	17	17

Table 4: RTAB Table for Example 3

4 Conclusion

A scheme was presented to provide scheduling guarantees for a variety of fault tolerant techniques. Bounds of execution were developed and an study case was examined to analyze these techniques in its ability to recover from transient faulty situations. A criterion for providing responsiveness to fault-tolerant scheduler were discussed and some approaches were developed. While the schedulability of the recovery workload has been addressed, the incorporation of criticalness to the workload remain to be solved. Simulation studies are underway to explore the potential of this approach for servicing a wide spectrum of recovery workloads.

References

- [1] Ghosh, S., "Guaranteeing Fault-Tolerant Through Scheduling in Real-Time Systems", *Ph.D. Thesis, Dep. Comp. Science, University of Pittsburgh*, 1997.
- [2] S.Ramos-Thuel and J.K.Strosnider, "The Transient Server Approach to Scheduling Time-Critical Recovery Operations", *Proc. the IEEE RTSS*, pp. 286-295, 1991.
- [3] J.W. Liu, and W.K. Shih, "Imprecise Computations", *Proceedings of the IEEE*, January, 1994.
- [4] J. Stankovic, "Advances in Real-Time Systems", *IEEE Computer Society Press*, April, 1992.
- [5] J.Lehoczky, L.Sha, Ye Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization...", *Proceedings of the IEEE RTSS*, pp. 166-171, 1989.