Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2014) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Information and Communication Technologies (ICICT 2014)

Fault Tolerant Scheduling of Mixed Criticality Real-Time Tasks under Error Bursts

Abhilash Thekkilakattil, Radu Dobrin and Sasikumar Punnekkat

Mälardalen University, Sweden

Abstract

Dependability is an important requirement in hard real-time applications due to the potentially catastrophic consequences of failures. In these systems, fault tolerance mechanisms like temporal redundancy are adopted to improve reliability. Most of these types of systems are increasingly moving towards integrating critical and non-critical functionalities on the same platform to, e.g., better utilize resources and further reduce cost, and are commonly deployed in environments where errors typically occur in the form of bursts e.g., due to Electro Magnetic Interference (EMI). Consequently, in mixed criticality real-time systems, the designer must guarantee that critical tasks are feasible even under the presence of the error burst, while ensuring the feasibility of the non-critical tasks that are not affected by the burst. We refer to this as *Fault Tolerance feasibility* (FT-feasibility) of mixed-criticality real-time systems.

In this paper, we build on the well established results on Earliest Deadline First (EDF) scheduling, to derive a sufficient test that determines the FT-feasibility of a set of mixed criticality real-time tasks under the assumption that the inter-arrival time between two consecutive error bursts is at least equal to the hyper-period of the taskset.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014).

Keywords: Mixed Criticality Systems, Real-time Systems, Error Bursts, Reliability

1. Introduction

Modern mission and safety critical real-time systems are moving towards integrating functionalities of different criticalities on the same platform to reduce Size Weight and Power (SWaP) constraints. These systems must be highly reliable, and especially, the critical part of the software needs to be tolerant towards faults. One way to guarantee the reliable and timely operation of the system is to use real-time fault tolerance mechanisms that can prevent failures. However, in order to reason about the correctness of the system, a realistic error model is required. Additionally, the overheads associated with the adopted fault-tolerance mechanisms must be considered. Traditionally, errors are

* Abhilash Thekkilakattil. Tel.: +46-21-101689.
E-mail address: abhilash.thekkilakattil@mdh.se

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014).

treated as singleton events which may not be realistic in situations where the errors occur continuously over a period of time, e.g., errors occurring on the computer system of a vehicle when it passes through an electromagnetic field. An error burst model, together with a suitable failure tolerance strategy, is more appropriate for such situations to provide feasibility guarantees to the failed tasks in mixed criticality systems.

The events occurring in a real-time system are mapped to a taskset comprising of different real-time tasks. Each task in the task set may generate an infinite number of jobs that need to finish executing by their associated deadlines. The real-time system may employ an appropriate fault tolerance strategy, such as temporal redundancy. Under temporal redundancy, the failed tasks are re-executed or an alternate task is executed before the associated deadline[1,2]. We refer to the original task executions as the *primary* of the task and the re-executions as *alternates* of the task. If the error burst occurs during the execution of alternates, recovery tasks are executed until one successful execution is achieved. However, from a feasibility perspective, these re-executions need to complete before the original task deadline in mixed criticality hard real-time systems. Previously, Many and Doose[3] and Aysan[4] proposed methods to determine the Fixed Priority schedulability of a set of real-time tasks under error bursts.

One of the main motivations behind mixed criticality scheduling is to enable the use of Commercial Off-the Shelf (COTS) hardware and software components while developing safety-critical and mission-critical real-time systems[5,6]. COTS components are typically developed by third parties, and hence may exhibit unpredictable behaviors at runtime that may jeopardize correctness (*w.r.t* value and timeliness) guarantees in the system. Therefore, while building safety and mission critical systems, the developer has to guarantee that the critical functionalities in the system does not fail even if the COTS components exhibit unpredictable behaviors. We focus our attention on providing timeliness guarantees to mixed criticality systems in the presence of error bursts using temporal redundancy. We define the Fault Tolerance feasibility (FT-feasibility) of a mixed criticality real-time system under an error burst of known upper bounded length, as the existence of a schedule that guarantees the successful re-execution of the critical jobs hit by the burst, as well as the feasible execution of non-critical jobs that are not hit by the burst.

In this paper, we consider the problem of determining the FT-feasibility of a set of temporally redundant *mixed criticality real-time tasks*. We build on the well established results on EDF to propose a *sufficient* test that determines whether any given mixed criticality real-time taskset is FT-feasible in presence of an error burst whose length is no more than a known upper-bound. For this purpose, we generalize our FT-feasibility analysis presented in[7] to the case of mixed criticality systems.

The rest of the paper is organized as follows: Section 2 details the system model and the problem definition, followed by our FT-feasibility analysis of mixed-criticality systems in section 3. before concluding in section 4.

2. System Model

In this Section, we describe the task model, error model, scheduling model as well as the notations used throughout this paper.

2.1. Fault and Error Model

A fault in a system is defined as the hypothesized cause of an error in the system[1]. We assume that all the faults occurring in a task causes an error, which in turn leads to its failure. An error in a task, that leads to its failure, can be seen as a fault in the system which can be tolerated by a suitable fault tolerance mechanism like temporal redundancy. Hence the use of the terms *faults*, *errors* and *failures* depend on the level of abstraction of the system that is considered. A majority of the related works in the literature treats error occurrences as singleton events. Instead, in this paper, we consider error bursts defined as a series of *soft/transient* or *intermittent* errors that occur within a known time interval that prevents successful task executions during that interval, e.g., radar waves in airport areas produce electromagnetic fields that can cause error bursts on computer systems[3,8]. The duration of the error burst depends on the duration for which the computer system is exposed to the EMI. We assume that the fault burst occurring during the task executions, that may affect the CPU, memory or the I/O subsystems, eventually manifests as an error burst on the tasks, leading to their failure. The underlying platforms such as operating system and the error detection mechanism is assumed to be resistant to the burst. Spatial redundancy techniques must be employed to tolerate such failures, which is outside the scope of this paper.

We assume that the upper-bound on the error burst length is known, and is denoted by T_{length}. Two consecutive error burst occurrences are assumed to be separated by a time duration no less than the LCM of the task periods.

2.2. Task Model

We assume a mixed criticality sporadic real-time taskset denoted by $\Gamma = \{\tau_1, \tau_2, ...\tau_n\}$. Consecutive jobs of each τ_i is assumed to be separated by a time interval no less than T_i, and has a Worst Case Execution Time (WCET) denoted by C_i and a relative deadline denoted by $D_i \leq T_i$. Every $\tau_i \in \Gamma$ belongs to either Γ_c or Γ_{nc}, where Γ_c is the subset of critical tasks and Γ_{nc} is the subset of non-critical tasks. A deadline miss on critical tasks can have catastrophic consequences, while a deadline miss on non-critical tasks does not cause any catastrophe, except for a minor degradation of system performance. Our aim is to guarantee the successful execution of all critical tasks even under the presence of error bursts, while guaranteeing the execution of all non-critical tasks that are not affected by the burst. In case a critical task is hit by the burst, they are re-executed or a recovery task is executed until a successful execution is achieved.

The tasks are assumed to be indexed according to the increasing order of relative deadlines. Each τ_i generates an infinite number of jobs, where the j^{th} job of τ_i is represented as $\tau_{i,j}$. The set of absolute deadlines of the taskset in the LCM (for the strictly periodic case) is denoted by $\Delta = \{d_1, d_2, ..., d_m\}$, $d_i < d_{i+1}$. Here, LCM is used to denote the hyper-period of the taskset. We also denote ϵ to represent a very small positive number arbitrarily close to zero.

2.3. Scheduling Model

We assume the earliest deadline based scheduler to determine the FT-feasibility of a set of mixed criticality real-time tasks under error bursts. The use of EDF enables us to leverage on the associated well established results [9] [10]. Our result facilitates the use of existing operating systems that support EDF, e.g., ERIKA [11], for mixed criticality fault tolerant scheduling of real-time tasks.

3. Schedulability Analysis

When a set of mixed criticality real-time tasks is hit by an error burst during any time interval, the failed tasks deviate from their specified behavior, e.g., gives the wrong output or overruns their WCET. This deviation in the behavior can be typically detected at the end of their budgeted time, and an alternate action can be taken by the scheduler such as the execution of an alternate task. In this time interval, between the last idle time in the system and the successful execution of all the failed tasks, there are three components that decide the mixed criticality schedulability:

1. The demand bound, which accounts for one correct execution of each task.
2. The duration of the error burst.
3. The time *wasted* in executing the failed tasks.

If a job $\tau_{i,j}$ is hit by a burst, a part of its execution time is *wasted* since the error detection happens at the end of its budgeted time. The wasted time is maximum if the error burst begins right before $\tau_{i,j}$ finishes its execution, or ends immediately after the primary/alternate of $\tau_{i,j}$ starts executing.

Definition 1. *The Maximum Wasted Execution Time (MWET) for a critical task τ_i that fails due to an error burst is defined as the maximum WCET of its failed primary or an alternate lying outside error burst.*

$$MWET(\tau_i) = C_i - \epsilon$$

An example of the MWET for a critical task is shown in Figure 1.

The error burst can affect jobs of multiple critical tasks leading to many such MWETs that waste the processor time. The sum of all the possible MWETs of all the critical tasks scheduled in a time interval of length t gives the *worst case temporal wastage* in t.

Definition 2. *The Worst Case Temporal Wastage (WCTW) in any time interval of length t, denoted by $W_{err}(t)$, is defined as the sum of MWETs of maximum possible aborted primaries and alternates of the critical tasks having their release times and deadlines in t.*

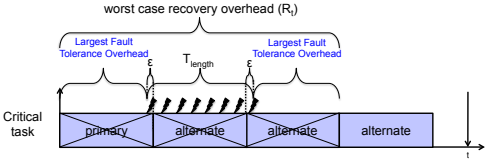

Fig. 1: Worst Case Error Overhead.

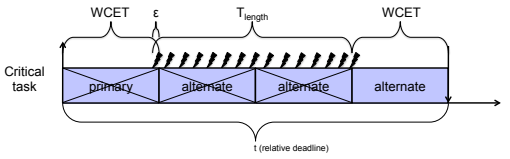

Fig. 2: Longest tolerable error burst.

In the following, we derive the equations to calculate $W_{err}(t)$ and derive a sufficient condition for the feasibility of a set of mixed criticality real-time tasks.

Outline of our approach: Our strategy for deriving the sufficient condition that determines the feasibility of a mixed criticality system is as follows. We consider a time instant t' when a job $\tau_{i,j}$, which can be critical or non-critical, having an absolute deadline at time t is executing on the processor. We assume that even if an error burst occurs in $[t', t]$ there are no deadline misses on the critical jobs scheduled in the interval $[t', t]$, and then derive a sufficient condition for this to be true. In the case of non-critical jobs, we assume that, if they are *not* hit by the error burst they are guaranteed a successful execution and derive a sufficient condition for this to be true. In order to avoid notational clutter, we assume that if there are more than one critical jobs with absolute deadline at t, $\tau_{i,j}$ denotes the critical job with the largest execution time. We also use t_0 to represent the earliest time instant before t' at which the processor is idle. Without loss of generality, we can set $t_0 = 0$. Here, our goal is to find the worst case temporal wastage in the interval $[t_0 = 0, t]$, *i.e.*, $W_{err}(t - t_0) = W_{err}(t)$.

The $W_{err}(t)$ occurs when the jobs of Γ arrive strictly periodically, because such a scenario maximizes the number of jobs that are hit by the error burst. We derive $W_{err}(t)$ for each of the following cases:

Case 1: No job scheduled in $[t', t]$ is hit by the error burst.

Case 2: Only one job scheduled in $[t', t]$ is hit by the error burst.

Case 3: At least two jobs scheduled in $[t', t]$ are hit by the error burst.

In the following, we show that if $\tau_{i,j}$ is a critical job, no job released in $[t', t]$ having an absolute deadline later than t are affected by the burst.

Observation 1. *If the error burst starts at t' and $\tau_{i,j}$ is a critical job, then no job $\tau_{a,b}$ released in $[t', t]$ with an absolute deadline greater than t is affected by the the error burst.*

Proof. The proof follows from the assumptions that a) there are no deadline misses even if some tasks are hit by the error burst, and b) the primaries and alternates are scheduled based on EDF. The job $\tau_{i,j}$ has a higher priority than $\tau_{a,b}$ because of its earlier absolute deadline. Consequently $\tau_{a,b}$ will execute only after $\tau_{i,j}$ has completed its execution. According to our assumption, the critical jobs are schedulable even if they are hit the error burst, and hence $\tau_{i,j}$ will eventually succeed no later than t. Therefore, when $\tau_{a,b}$ starts its execution, the error burst would have ended since the critical job $\tau_{i,j}$ would have completed one successful execution. Therefore (any such) $\tau_{a,b}$ is not affected by the burst. □

Lemma 1. *If no job is released in $[t', t]$ with absolute deadline $\leq t$, the WCTW $W_{err}(t)$ is given by:*

$$W_{err}(t) = \begin{cases} 2(C_i - \epsilon), & if\ \tau_i \in \Gamma_c \\ 0, & otherwise \end{cases}$$

Proof. If $\tau_{i,j}$ is critical, its primary or an alternate has to finish one successful execution for any job other than $\tau_{i,j}$ to start its execution. Thus, as seen from Observation 1, $\tau_{i,j}$ is the *only* job that fails due to the error burst.

Since we consider mixed criticality systems, we need to provide re-execution guarantees to only the critical jobs. Therefore $W_{err}(t)$ occurs when $\tau_{i,j}$ is a critical job, and the error burst starts ϵ units before its primary finishes executing, and ends ϵ units after its last failed alternate starts execution (an example is given in Figure 1). The value of $W_{err}(t)$ is

$$W_{err}(t) = 2(C_i - \epsilon)$$

If $\tau_{i,j}$ is non-critical, since we consider a scenario where no job with deadline $< t$ is released in $[t', t]$, then some task with a deadline greater than t can fail due to the error burst. This is because we assume that non-critical jobs are not re-executed upon failures. Once the budgeted time of the failed primary of $\tau_{i,j}$ expires, a job with a deadline greater than t can be scheduled in $[t', t]$. However, these failed primaries and/or alternates will influence the WCTW at some time instant *later* than t. Therefore, in this case,

$$W_{err}(t) = 0$$

The proof follows by unifying the above two equations. □

Observation 2. *The relative deadline of every job with a release time in* $[t', t]$*, and an absolute deadline* $< t$*, is less* D_i*.*

This is quite straightforward as $\tau_{i,j}$ has been released prior to time instant t'. Consequently, every job released in $[t', t]$ with an absolute deadline less than t will have a relative deadline less than that of τ_i. We now consider case 1 in which no job scheduled in $[t', t]$ is hit by the burst.

Lemma 2. *If an arbitrary job* $\tau_{a,b}$ *with an absolute deadline given by* $d_l = bT_a + D_a$ *is not affected, and the error burst started at some time prior to its start time,*

$$W_{err}(d_l) = W_{err}(d_{l-1})$$

Proof. In this case, some higher priority critical jobs that executed prior to the start of $\tau_{a,b}$ are hit by the error burst. Let the d_k denote the latest deadline of such a high criticality job prior to d_l. In this case, the $W_{err}(d_k)$ can cause a deadline miss at d_l. Therefore,

$$W_{err}(d_l) = W_{err}(d_k)$$

There could be more jobs having an absolute deadline in the interval between d_k and d_l. These jobs are not affected by the error burst since it has already ended at d_k. The WCTW at all these deadlines is equal to $W_{err}(d_k)$.

$$\Rightarrow W_{err}(d_l) = W_{err}(d_{l-1})$$

□

In the following, we consider the case when only a single job fails due the burst in $[t', t]$.

Lemma 3. *If only a single job fails due to the error burst in* $[t', t]$*, the* $W_{err}(t)$ *is given by:*

$$W_{err}(t) = \begin{cases} \max\{2(C_k - \epsilon)\},\ \forall \tau_k \in \Gamma_c : D_k \leq D_i \\ 0, \qquad\qquad\qquad\qquad\quad otherwise \end{cases}$$

Proof. We showed earlier that all the jobs with release times and deadlines in $[t', t]$ are jobs with relative deadline less than that of τ_i. If a job of $\tau_k \in \Gamma_c$ executing in the interval $[t', t]$ is hit, the contribution is twice the maximum of the corresponding MWETs. In this case, the error burst starts an arbitrarily small time unit before the failed primary of the critical task finishes executing and ends an arbitrarily small time unit after the last failed alternate has started executing (proved in Lemma 1). Here, τ_k can be either τ_i, if it is critical, or, as per observation 2, any critical task τ_a such that $D_a \leq D_i$.

The contribution to $W_{err}(t)$ is equal to zero if τ_i is non-critical, and there are no critical jobs having a relative deadline less than or equal to D_i. The proof follows. □

In the next Lemma, we find an upper-bound on the contribution of job $\tau_{i,j}$ to $W_{err}(t)$ when more than one jobs are hit by the error burst in $[t', t]$.

Lemma 4. *If the error burst leads to the failure of at least one more job in addition to* $\tau_{i,j}$*, in the interval* $[t', t]$*, the contribution of* $\tau_{i,j}$ *to* $W_{err}(t)$ *at time* t *happens when* $\tau_{i,j}$ *is a critical job and the contribution is equal to* $2(C_i - \epsilon)$*.*

Proof. According to our assumptions, only the critical jobs are re-executed upon failures, and hence $\tau_{i,j}$ contributes to $W_{err}(t)$ only if it is a critical job, otherwise its contribution to $W_{err}(t) = 0$.

The contribution of $\tau_{i,j}$ is maximum when its primary is hit ϵ units before completion, and one of its failed alternates is preempted after an arbitrarily small time duration (ϵ units) after the start of execution. The contribution of the primary of $\tau_{i,j}$ to $W_{err}(t)$ is equal to $(C_i - \epsilon)$.

Since the alternate of $\tau_{i,j}$ can be preempted by one or more jobs (in a nested manner), we have two cases- 1) none of the preempting jobs are critical 2) at least one of the preempting job is critical.

Case 1: No jobs preempting $\tau_{i,j}$ are critical.

If only non-critical jobs preempt $\tau_{i,j}$, all these jobs will be affected until the error burst ends. However, they do not contributed to $W_{err}(t)$. When $\tau_{i,j}$ resumes its execution, the worst case contribution of $\tau_{i,j}$ occurs a) when the error burst ends before $\tau_{i,j}$ resumes its execution or b) when the error burst ends just after the last failed alternate of $\tau_{i,j}$ has started execution. The contribution of $\tau_{i,j}$ in both the cases a and b is at most $(C_i - \epsilon)$.

Case 2: At least one job preempting $\tau_{i,j}$ is critical.

According to our assumption, all critical jobs are feasible even if there are error bursts. Hence, all critical jobs that are preempting $\tau_{i,j}$ in a nested manner, will complete its execution successfully before $\tau_{i,j}$ is again scheduled. Hence the error burst would have ended when $\tau_{i,j}$ resumes its execution. Consequently, the remaining execution time of $\tau_{i,j}$, i.e., $C_i - \epsilon$, is wasted as it was hit by the burst before being preempted. Since the error burst has already ended, the alternate of job $\tau_{i,j}$ can execute successfully. The contribution of the failed alternate of $\tau_{i,j}$ to the $W_{err}(t)$ is $(C_i - \epsilon)$.

Therefore, the contribution of $\tau_{i,j}$ to $W_{err}(t)$ at time t is equal to $2(C_i - \epsilon)$. □

No more than one primary or alternate of the critical jobs executing in the interval $[t', t]$ contributes to $W_{err}(t)$.

Lemma 5. *If the error burst leads to the failure of at least one more job in addition to $\tau_{i,j}$, in $[t', t]$, no more than one affected primary or alternate of each critical job other than $\tau_{i,j}$ will contribute to $W_{err}(t)$.*

Proof. We had assumed that the job $\tau_{i,j}$ is the first to be affected by the burst, and it contributes to $W_{err}(t)$ *only* if it is a critical job. Its contribution, in this case, is equal to $2(C_i - \epsilon)$, as presented in the previous Lemma.

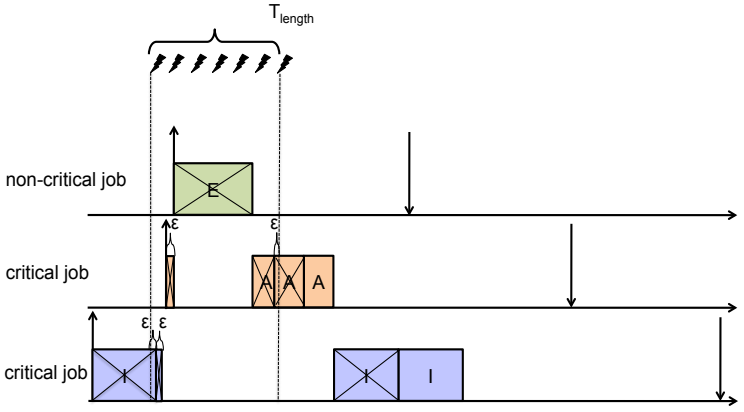

Fig. 3: Critical and non-critical jobs under an error burst in a mixed criticality system

Consider a job $\tau_{a,b}$, that is the next job to be hit by the burst. The job $\tau_{a,b}$ has a release time and deadline in the interval $[t', t]$. If $\tau_{a,b}$ is a critical job, then it must recover before its absolute deadline for the task set to be feasible. The contribution of $\tau_{a,b}$ to $W_{err}(t)$ is maximum when $\tau_{a,b}$ is a critical job and one of the following is true:

Case 1: The failed primary or an alternate of $\tau_{a,b}$ is preempted after an arbitrarily small duration from the start of its execution, by a higher priority job $\tau_{e,f}$.

If $\tau_{e,f}$ is a critical job, it will complete one successful execution and the error burst will end prior to its completion. After it completes, the failed $\tau_{a,b}$, which was preempted, executes wasting the processor time. Thus, according to the definition 1, the largest processor time wasted by $\tau_{a,b}$ before it can successfully execute is $(C_a - \epsilon)$. On the other hand, if $\tau_{e,f}$ is non-critical, it need not achieve one successful execution before its deadline. However, the MWET by $\tau_{a,b}$ at time t is still $(C_a - \epsilon)$, which is when the error burst ends right after the last failed alternate of $\tau_{a,b}$ starts executing (see Figure 3).

Case 2: The error burst ends right before the start of the last affected alternate of $\tau_{a,b}$.

In this case $\tau_{a,b}$ is the only critical job in addition to $\tau_{i,j}$ that fails due to the burst. If $\tau_{a,b}$ is a critical job, the contribution of $\tau_{a,b}$ to $W_{err}(t)$ is $(C_a - \epsilon)$, before $\tau_{a,b}$ completes without a failure.

Hence, if $\tau_{a,b}$ is a critical job, its contribution to $W_{err}(t)$ is $(C_a - \epsilon)$. The argument presented above can be recursively applied for all higher priority critical jobs $\tau_{e,f}$ that are released between the release time and deadline of $\tau_{a,b}$. An example for the case where more than one job is hit is given in Figure 3.

Thus we can conclude that if at least two jobs are hit by the error burst, no more than one affected primary or an affected alternate of each *critical* job other than $\tau_{i,j}$ will contribute to $W_{err}(t)$. □

We now derive the $W_{err}(t)$ when at least two jobs are hit by the error burst in the interval $[t', t]$, corresponding to the case 3 enumerated above.

Lemma 6. *If the error burst leads to the failure of at least one more job in addition to $\tau_{i,j}$, in the interval $[t', t]$, the WCTW $W_{err}(t)$ is given by:*

$$W_{err}(t) = \begin{cases} 2(C_i - \epsilon) + \sum\limits_{\forall \tau_k \in \Gamma_c : D_k < D_i} (C_k - \epsilon), & if\ \tau_i \in \Gamma_c \\ \sum\limits_{\forall \tau_k \in \Gamma_c : D_k < D_i} (C_k - \epsilon), & otherwise \end{cases}$$

Proof. It is evident from Lemma 5 that no more than one primary or one alternate of the affected *critical* jobs with releases and deadlines in $[t', t]$ will contribute to $W_{err}(t)$. Hence the total contribution to $W_{err}(t)$ is largest when *every critical job* $\tau_{a,b}$ released in the interval $[t', t]$, with a deadline before t, fails and contributes $C_a - \epsilon$ time units to $W_{err}(t)$. Such a case occurs when the jobs in the interval $[t', t]$ preempt each other in a nested manner, with the preemption on a critical job occurring ϵ units after its primary or alternate starts executing.

From observation 2, we know that the jobs that are released in $[t', t]$ have relative deadlines less than D_i. The contribution of these jobs to the WCTW is $\sum_{\forall \tau_k \in \Gamma_c : D_k < D_i} (C_k - \epsilon)$. According to Lemma 4, the contribution of $\tau_{i,j}$ to $W_{err}(t)$ is $2(C_i - \epsilon)$, if it is a critical task, and 0 otherwise. The proof follows by summing up the respective contributions. □

In the following, we derive $W_{err}(t)$ and use this in Theorem 2 to derive the condition for FT-feasibility.

Theorem 1. *In a mixed criticality system, the worst case temporal wastage $W_{err}(t)$ where $t = d_l = jT_i + D_i$ for any job $\tau_{i,j}$, is given by:*

$$W_{err}(t) = \max\left(x, y, W_{err}(d_{l-1})\right)$$

Here,

$$x = \begin{cases} 2(C_i - \epsilon) + \sum\limits_{\forall \tau_k \in \Gamma_c : D_k < D_i} (C_k - \epsilon), & if\ \tau_i \in \Gamma_c \\ \sum\limits_{\forall \tau_k \in \Gamma_c : D_k < D_i} (C_k - \epsilon), & otherwise \end{cases}$$

$$y = \begin{cases} \max\{2(C_k - \epsilon)\}, \forall \tau_k \in \Gamma_c : D_k \leq D_i \\ 0, & otherwise \end{cases}$$

Proof. The proof follows from Lemma 2, 6, 3. The $W_{err}(t)$, where $t = jT_i + D_i$ for any $\tau_{i,j}$ is obtained from by the max of the $W_{err}(t)$ given by Lemmas 2, 3 and 6. □

We now define the *worst case error overhead* at any time instant t introduced by our mixed criticality model.

Definition 3. *The worst case error overhead E_t at an arbitrary time instant t is defined as*

$$E_t = T_{length} + W_{err}(t)$$

Figure 1 illustrates the worst case error overhead at time instant t i.e., E_t. Building on the demand based analysis by Baruah et al.[9], we can obtain a sufficient condition for FT-feasibility of mixed-criticality real-time systems.

Theorem 2. *A mixed criticality real-time task set Γ is FT-feasible, such that the tasks are hit by an error burst of length at most T_{length}, if, $\forall t = kT_j + D_j, \forall \tau_j \in \Gamma$ and $t \leq LCM$,*

$$E_t + \sum_{i=1}^{n} DBF_i(t) \leq t$$

Proof Sketch. Assume that the condition presented above does not hold for some t, i.e., $W_{err}(t) + \sum_{i=1}^{n} DBF_i(t) > t - T_{length}$. In this case, during the interval of length t, the demand bound requested by the real-time taskset is greater than the available processor time outside the error burst. We do not present the formal proof which is similar to the one presented by Aydin[2]. □

The Theorem 2 is only a sufficient test for determining feasibility of a mixed criticality real-time task set. This is because, in many cases, the actual WCTW at t may not be as high as the one calculated by our method.

4. Conclusions

In this paper, we presented a test to determine the fault tolerance feasibility of mixed criticality real-time tasks affected by an error burst of known upper-bounded length. We have assumed that the underlying scheduler is the Earliest Deadline First scheduler that has enabled us to bild on the associated schedulability tests. The test that we have proposed enables a system designer to determine the schedulability of a given set of mixed criticality real-time tasks that are affected by error bursts. Future work will include extensions of the approach to different error models as well as to multiprocessing environments.

Acknowledgment

This work was supported by the Swedish Research Council project Contesse (2010-4276).

References

1. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable Secure Computing* 2004;.
2. Aydin, H.. Exact fault-sensitive feasibility analysis of real-time tasks. *IEEE Transactions on Computers* 2007;.
3. Many, F., Doose, D.. Scheduling analysis under fault bursts. In: *The 17th IEEE Real-Time and Embedded Technology and Applications Symposium*. 2011, .
4. Aysan, H.. Fault-tolerance strategies and probabilistic guarantees for real-time systems. In: *PhD thesis, Malardalen University*. 2012, .
5. Vestal, S.. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In: *The 28th IEEE International Real-Time Systems Symposium, 2007*. 2007, .
6. Burns, A., Baruah, S.. Timing faults and mixed criticality systems. In: *Dependable and Historic Computing*; Lecture Notes in Computer Science. 2011, .
7. Thekkilakattil, A., Dobrin, R., Punnekkat, S., Aysan, H.. Resource augmentation for fault-tolerance feasibility of real-time tasks under error bursts. In: *The 20th International Conference on Real-Time and Network Systems*. ACM; 2012, .
8. ARP5583A, . Guide to certification of aircraft in a high-intensity radiated field (HIRF) environment. In: *Ae-4 Electromagnetic Environmental Effects (E3) Committee, SAE International, Product Code: ARP5583A, Revision Number: A*. 2010, .
9. Baruah, S.K., Rosier, L.E., Howell, R.R.. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems* 1990;.
10. Baruah, S., Mok, A., Rosier, L.. Preemptively scheduling hard-real-time sporadic tasks on one processor. In: *The 11th Real-Time Systems Symposium*. 1990, .
11. Cirinei, M., Mancina, A., Cantini, D., Gai, P., Palopoli, L.. An educational open source real-time kernel for small embedded control systems. In: *Computer and Information Sciences*. Springer Berlin / Heidelberg; 2004, .