Scheduling Anomaly Detection and Optimization for Distributed Systems with Preemptive Task-Sets

Razvan Racu, Rolf Ernst
Institute of Computer and Communication Network Engineering
Technical University of Braunschweig
D-38106 Braunschweig, Germany
{racu|ernst}@ida.ing.tu-bs.de

Abstract

Robustness and optimality are becoming the key principles in designing efficient and reliable state-of-the-art multi-processor real-time systems. However, due to complex inter-processor dependencies, the variation of local system parameters may have unpredictable system level impact including timing anomalies. In this context, the former heuristic optimization approaches used at resource level become less suitable for distributed systems with heterogeneous components and dynamic scheduling techniques. Techniques from exploration theory can better address the optimization problems but suffer from huge search spaces in general. In this paper, we present constructive methods for pointing out those system configurations that lead to anomalous behavior of the performance metrics. These are then used to guide the exploration process and reduce the search space, thereby increasing efficiency and making the approach applicable in practice. As a result, detailed information about anomalies can be quickly obtained and heavily exploited in system optimization, which we demonstrate using comprehensible examples.

1. Introduction

As a current trend, traditional single-processor architectures are replaced by large heterogeneous, networked systems based on multi-processor systems-on-chip (MpSoC) in order to run complex distributed applications with real-time constraints.

As a consequence of this development towards concurrency on multiple levels of an architecture, the system designer faces several new challenges concerning design, integration and performance verification of such systems. The entire design process became more complicated due to real-world design flows characterized by tight time-to-market, permanently changing requirements and very complex supply-chains including platform based design, subsystem integration and IP protection and reuse. Therefore, in the early design phases of such systems not all information required by the performance analysis is available up front. Instead, designers must consider incomplete specifications, early performance estimates, asserted values for different system parameters, and so on. In this context, design space exploration and system optimization are two issues intensively debated by the real-time community. Sophisticated system dependencies can easily turn subsystem best-case performance into system worst-case performance. Therefore, the former heuristic techniques used for resource level optimization become less suitable.

In case of functional dependencies between tasks on different system components, the performance metrics may have unpredictable, anomalous behavior. A designer should certainly be aware of potential anomaly hazards and their quantitative effects to find optimal and robust system configurations. Therefore, detecting the system parameter values that lead to such anomalies is a key problem in controlling the design space exploration and optimization process.

In this paper, we investigate preemptive task sets and the influence of the activation offsets on the behavior of different performance metrics, like task response times or path latencies. Furthermore, we provide an efficient constructive framework to calculate the bounds of the behavioral intervals. Compared to time-consuming curve-traversal approaches, constructive methods substantially help guiding the exploration process and reducing the search space during optimization. As a result, detailed information about anomalies can be quickly and reliably obtained.

After an overview on the state of the art in Section 2, Section 3 shows that modifications of the task activation offsets dynamically change the critical instant of the lower priority tasks, leading to unpredictable, anomalous behavior of their latencies. In Section 4 we bounds the anomaly

intervals characterizing the worst-case scenarios. Section 5 describes the algorithm used to identify the task-pairs with anomalous dependencies. Finally, the application and the benefits of the proposed algorithms are demonstrated using a synthetic system example.

2. State-of-the-Art in Performance Analysis

System performance validation is key during the design of state-of-the-art real-time distributed systems. With the increasing system complexity new performance analysis models are required. Therefore, schedulability analysis has grown beyond the single processor platforms and can now cover large heterogeneous systems.

The *holistic analysis* approach developed by Tindell [16] has extended former local analysis techniques, considering the scheduling influences along functional paths in the system. He proposed a performance verification model for distributed real-time systems with preemptive task sets communicating via message passing and shared data areas. Pop *et al.* [11] extended this approach for systems consisting of fixed-priority scheduled CPUs connected via a TDMA scheduled bus.

The *compositional analysis* approach combines local scheduling analysis and event model propagation into a system-level analysis. Richter [14] proposed a compositional analysis model based on event model interfaces. Jersak [4] has extended this approach to allow performance analysis of applications with complex task dependencies, including multiple activating inputs, functional cycles and multi-rate data dependencies with intervals. Chakraborty *et al.* [1] proposed a similar compositional approach based on real-time calculus.

In general, system level analyses are based on classical schedulability tests and algorithms available for the analysis of local system components. Well known are the analyses for static priority preemptive scheduling from Liu and Layland [8] and Lehoczky [7]. Similar algorithms were derived for time-driven [6] and deadline-driven [3] arbitration techniques. However, some of these analyses can be unnecessarily pessimistic, because they ignore certain correlations between tasks, leading to overly pessimistic worst-case load distribution over time. Such an example is the analysis of the task set mapped on CPU, as shown in Figure 1. Since the analysis assumes that all tasks are independent, all tasks are released simultaneously at the critical instant. In reality, this may never happen because τ_1 and τ_2 have correlated activation times. Such *precedence relations* were used by Tindell [15], Palencia [9, 10] and Henia [5] to compute tighter bounds for the response time analysis.

Recently, important work has been done to calculate best-case response time bounds [2, 13], and to reduce the timing uncertainty at task completion.

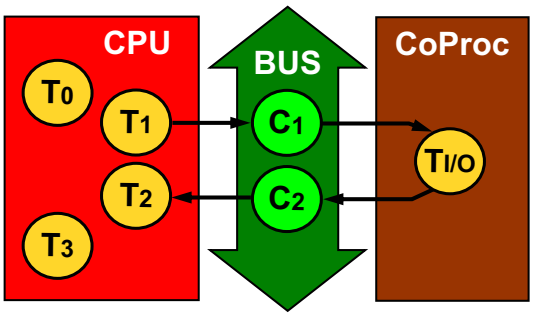


Figure 1. System with precedence relations

Other system-level performance analysis models are based on timed-automata, hybrid-automata, timed Petri-nets, synchronous languages, temporal logic, binary decision diagrams etc. However, they are not described in detail in this section since they use different formalisms from that used in this paper.

An important aspect that should be noticed is that all performance analysis models require that all timing properties of the system are completely specified. The system feasibility is decided by comparing the calculated performance metrics against the set of timing constraints. However, if not all constraints are satisfied, then it is often difficult to determine the source of the problem. This is due to the complex dependencies between system configuration and performance. The approach proposed in this paper efficiently points out system configurations leading to anomalous behavior of the performance metrics. It can be shown that variations between these points are safe with respect to predictability of system performance.

In the next section we show that variations of task execution times may lead to anomalous behavior of different system timing properties, and we determine the intervals where the performance metrics have a predictable behavior. In this paper we focus only on preemptive task sets. Similar anomalies can be found also for task sets mapped on resources with other scheduling strategies, such as Earliest Deadline First (EDF) or Round Robin.

3. Problem Formulation

The schedulability tests introduced by Liu and Layland [8] check the feasibility of a preemptive task set assuming that the tasks are periodically activated and do not interfere with each other. They defined the *critical instant* of a task as the instant at which a request for that task would have the largest response time. They proved that the critical instant of any task occurs whenever the task is released simultaneously with releases of all higher priority tasks. Based on this scenario the response time analysis calculates the worst-case response time of the task. Later on, Lehoczky [7] extended the response time analysis for task

sets with arbitrary deadlines and more complex task activation models. However, the assumption that the tasks do not communicate with each other was preserved, and the definition of the *critical instant* was preserved, too.

With the appearance of complex multi-processor distributed systems new analysis models were required. The classical schedulability tests used for single-processor architectures became unnecessarily pessimistic. Ignoring the functional dependencies between tasks, the response time analysis determines too conservative results. Therefore, the definition introduced by Liu and Layland for the critical instant slightly changed, leading to more complex worst-case execution scenarios. Tindell [15] and Palencia [9, 10] considered the correlations between the release times of different tasks in order to compute tighter response time bounds.

Consider, for example, the system shown in Figure 1. Task T_1 is periodically executed on the processing element CPU. It sends data at completion over the communication element BUS at the dedicated processor CoProc, handling for example external I/O operations. After data is completely transmitted, the coprocessor sends a status message back to CPU. The message activates the execution of task T_2. In addition to T_1 and T_2, two periodic tasks, T_0 and T_3 are executed on CPU. The arbitration of CPU is carried out according to a static priority preemptive scheduling policy. On BUS, channels C_1 and C_2 transfer data according to a fixed communication order, such that they can not disturb each other. Their response times are defined by their communication times. In this example, C_1 and C_2 have the communication times $C_{\mathsf{C}_1} = 5$ and $C_{\mathsf{C}_2} = 5$. The I/O subroutine, $\mathsf{T}_{\mathsf{I/O}}$ running on the dedicated coprocessor has a constant execution time, $C_{\mathsf{T}_{\mathsf{I/O}}} = 10$. The tasks mapped on CPU have the following execution demands: $C_{\mathsf{T}_0} = 10$, $C_{\mathsf{T}_1} = 30$, $C_{\mathsf{T}_2} = 25$ and $C_{\mathsf{T}_3} = 150$. The tasks are periodically activated with the following periods: $\mathcal{P}_{\mathsf{T}_0} = 130$, $\mathcal{P}_{\mathsf{T}_1} = \mathcal{P}_{\mathsf{T}_2} = 200$ and $\mathcal{P}_{\mathsf{T}_3} = 500$.

Since the execution of T_2 functionally depends on the execution of T_1, obviously there exists a timing correlation between the activations of T_1 and T_2. Therefore, the definition of the *critical instant* introduced in [8] may not apply in all situations. In [15], Tindell showed that the *critical instant* of T_3 occurs either at the instant when T_3 is released at the same time with T_1, or at the instant when T_3 is released at the same time with T_2. The worst-case response time of T_3 depends, on one hand, on the execution demands of T_1 and T_2, and on the other hand, on the offset between the activations of T_1 and T_2.

Figure 2(a) shows the Gantt-charts corresponding to worst-case execution scenario of T_3. At its critical instant T_3 occurs at the same time with the activation of T_1, and completes its execution after $R_{\mathsf{T}_3} = 290$. Task T_2 is released after a timing offset $\phi_2 = 50$. This offset represents the minimum time span between the activation of T_1 and

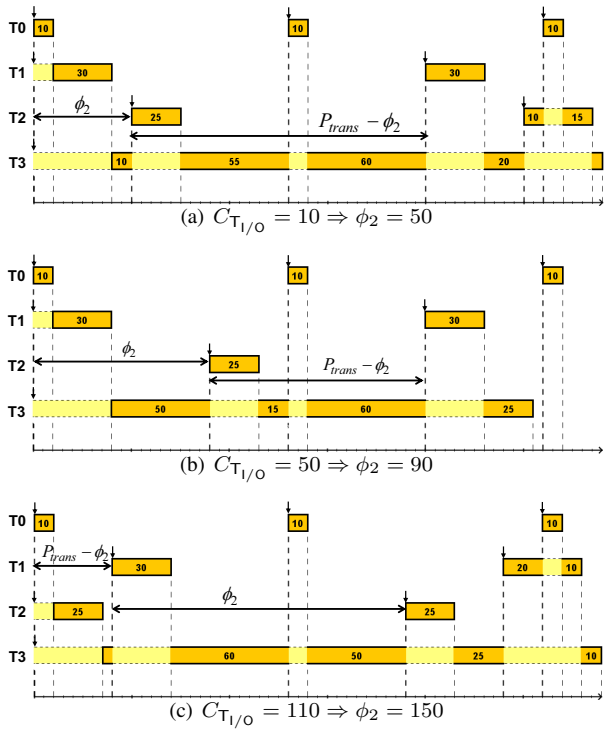


Figure 2. Variation of WCRT of T_3 due to modifications of the offset between T_1 and T_2

the activation of T_2, and, for the example in Figure 1 is computed as the sum of best-case response times of T_1, C_1, $\mathsf{T}_{\mathsf{I/O}}$ and C_2.

Consider that the I/O subroutine running on the coprocessor changes such that $\mathsf{T}_{\mathsf{I/O}}$ would have an execution demand $C_{\mathsf{T}_{\mathsf{I/O}}} = 50$. The delay between T_1 and T_2 becomes larger, and the worst-case scheduling scenario of T_3 changes, as shown in Figure 2(b). Although $C_{\mathsf{T}_{\mathsf{I/O}}}$ increased, the worst-case response time of T_3 decreased from 290 to 255. The new value of ϕ_2 shifted the activations of T_2 such that only one preemption from T_2 occurred during the worst-case response time of T_3. Moreover, the interference from other higher priority tasks, for example T_0, was reduced, as well.

Again, the execution demand of $\mathsf{T}_{\mathsf{I/O}}$ changes to $C_{\mathsf{T}_{\mathsf{I/O}}} = 110$. Figure 2(c) shows the worst-case execution scenario of T_3 corresponding to the new delay between T_1 and T_2, $\phi_2 = 150$. We can observe that, a further increase of $C_{\mathsf{T}_{\mathsf{I/O}}}$, and thus ϕ_2, shifted the activation of T_2 close to the next activation of T_1. Therefore, at the critical instant, T_3 is released simultaneously with T_2 and not with T_1. Because T_1 is released right after T_2, the interference from T_0, T_1 and T_2 during the response time of T_3 is the same as in the first case (Figure 2(a)).

As the simple case study shows, modifying the offset between the activation of two tasks mapped on the same

resource may lead to different response times of all lower priority tasks mapped on the same resource. In general, the worst-case response time of a lower priority task belongs to one of the behavioral intervals presented in Figure 3.

Since this offset is determined by the best-case response times of all tasks located within the same dependency path, such anomalies may occur modifying the execution demands of any of these tasks. Obviously, the bounds of the behavioral intervals strongly depend on the configuration of the entire task set.

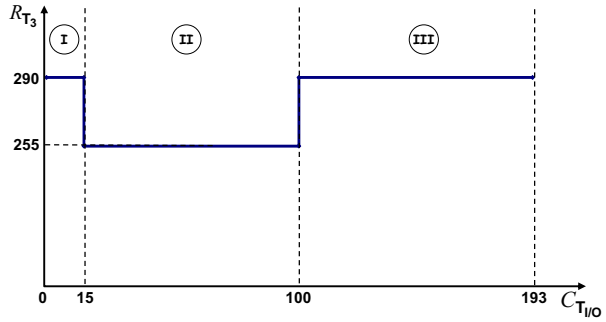


Figure 3. The worst-case response time of τ_3 as function of the execution demand of $\tau_{I/O}$

The variation of the offset between the activations of two tasks mapped on the same resource may influence not only the worst-case response time of the lower priority tasks, but the scheduling of the offset dependent task, as well. Consider again the system presented in Figure 1. The tasks have the same configuration as described above. Initially, the execution demand of the I/O routine is $C_{\tau_{I/O}} = 30$. The offset between the activations of τ_1 and τ_2 equals to $\phi_2 = 70$. Figure 4(a) shows the scheduling diagrams corresponding to worst-case response time of τ_2. As it can be observed, no interference occurred from τ_1 during the execution of τ_2. τ_2 completes its execution after $R_{\tau_2} = 35$.

If the processing time of $\tau_{I/O}$ increases to $C_{\tau_{I/O}} = 130$, the execution of τ_2 is preempted by next activation of τ_1, as shown in Figure 4(b). Task τ_2 experiences its maximal worst-case response time $R_{\tau_2} = 65$.

If the execution time of $\tau_{I/O}$ is again modified to $C_{\tau_{I/O}} = 180$, then the offset between the requests of τ_1 and τ_2 becomes $\phi_2 = 220$. As shown in Figure 4(c), τ_2 is released during the next activation of τ_1. τ_2 is blocked before execution only by the non-executed part of τ_1. Therefore, the worst-case response time of τ_2 decreased to $R_{\tau_2} = 45$. Moreover, it can be easily observed that the worst-case response time of τ_2 linearly decreases with the increase of ϕ_2. For values of ϕ_2 larger than $\mathcal{P}_{\tau_1} + r_{\tau_1}$, the worst-case scheduling of τ_2 is similar to the case $(\phi_3 \bmod \mathcal{P}_{\tau_1}) + r_{\tau_1}$, where r_{τ_1} represents the best-case response time of τ_1.

The above case study showed that the worst-case response time of τ_2 may belong to three behavioral intervals

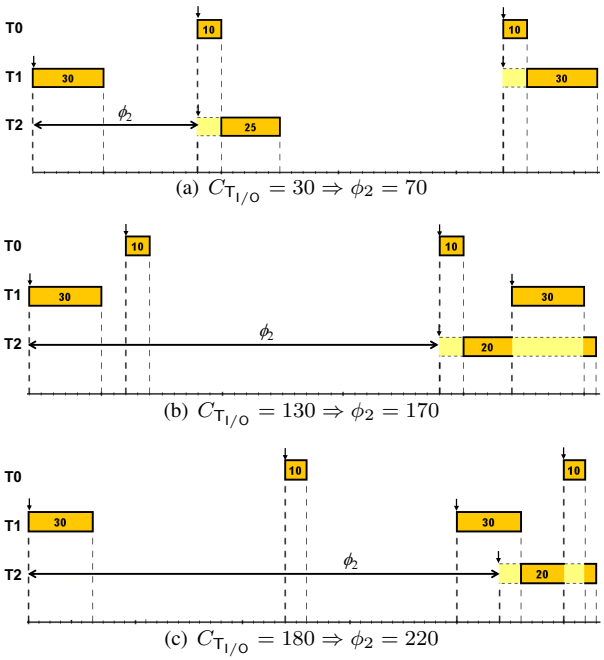


Figure 4. Variation of WCRT of τ_2 due to modifications of the offset between τ_1 and τ_2

depending on the value of its activation offset. The intervals are presented in Figure 5.

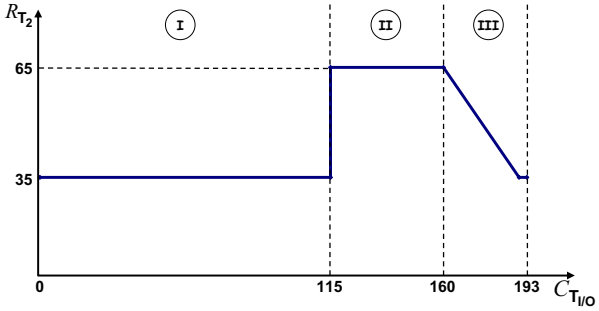


Figure 5. The worst-case response time of τ_2 as function of the execution demand of $\tau_{I/O}$

Since the system path latencies directly depends on the response times of the individual tasks within the paths, it is obvious that the anomalous behavior of the task response times determines also unpredictable behavior of the end-to-end timing properties. Increasing the execution/communication times of a task/channel directly increases its response time, but, at the same time, can reduce the response times of other tasks in the system, and therefore the path latencies. Finding out the values leading to anomalous behavior is an essential issue for the exact characterization of all performance metrics.

In Section 4 we propose an algorithm to determine the

values of the activation offset bounding the different behavioral intervals of the response time.

4. Bounds of Behavioral Intervals

In this section we present a detailed analysis of the behavioral interval bounds described in Section 3. The analysis is based on the worst-case response time equations derived for the schedulability tests of preemptive task sets with hard real-time constraints. The algorithm determines the values of the activation offsets leading to particular worst-case response times. Since the activation offset depends on the best-case response time of the task within the dependency path, a straight-forward algorithm can be carried out to determine the task execution demands corresponding to the calculated offsets.

4.1. Tasks with Independent Activations

Consider a set of n periodic independent tasks mapped on a resource and executed according to a static priority preemptive arbitration policy. Task i, denoted τ_i, has period $\mathcal{P}_i$, and execution time C_i. The tasks priorities are assigned in reverse order with respect to their indices, i.e. $prio(\tau_i) > prio(\tau_j)$ if $i < j$. According to [7], the worst-case response time of τ_i can be calculated using the following equation:

$$R_i^{(n+1)} = C_i + \sum_{\forall j \in HP(i)} \left\lceil \frac{R_i^{(n)}}{\mathcal{P}_j} \right\rceil \cdot C_j; \quad R_i^{(0)} = C_i \quad (1)$$

where $HP(i)$ contains the indices of all tasks with priorities higher than τ_i. The worst-case response time of τ_i, denoted R_i, is iteratively calculated using (1), until $R_i^{(n+1)} = R_i^{(n)}$.

According to [15], if additional inter-task dependencies exist, the worst-case response time analysis changes. Assume that tasks τ_k and τ_m belong to the same transaction $trans$. A transaction is a collection of tasks related either through some collectively performed function, or through some shared timing attributes whereby it is convenient to collect these tasks into a single entity [15]. If $prio(\tau_i) < prio(\tau_k)$ and $prio(\tau_i) < prio(\tau_m)$, then the analysis of the worst-case response time of τ_i is carried out analyzing two scenarios that may generate the worst-case. In the first case, the *critical instant* of τ_i is determined by the simultaneous activation of τ_i, τ_k and all tasks τ_j such that, $j \in HP(i)$ and $\tau_j \notin trans$. In the second scenario, the *critical instant* of τ_i is determined by the simultaneous activation of τ_i, τ_m and all tasks τ_j such that, $j \in HP(i)$ and $\tau_j \notin trans$. Obviously, since $\tau_k, \tau_m \in trans$, the periods $\mathcal{P}_k$ and $\mathcal{P}_m$ are equal to transaction period, $\mathcal{P}_{trans}$. Additionally, we denote by ϕ the minimum offset between two consecutive activations of τ_k and τ_m. Depending on the value of ϕ, the worst-case response time analysis selects, out of the two scenarios, the one building the longest response time of τ_i. For the first scenario, Equation (1) changes into

$$\begin{aligned} R_i'^{(n+1)} &= C_i + \sum_{j \in HP(i)}^{j \neq m} \left\lceil \frac{R_i'^{(n)}}{\mathcal{P}_j} \right\rceil \cdot C_j \\ &\quad + \left\lceil \frac{R_i'^{(n)} - \phi}{\mathcal{P}_{trans}} \right\rceil \cdot C_m; \quad R_i'^{(0)} = C_i \end{aligned} \quad (2)$$

and for the second scenario the worst-case response time of τ_i is determined by

$$\begin{aligned} R_i''^{(n+1)} &= C_i + \sum_{j \in HP(i)}^{j \neq k} \left\lceil \frac{R_i''^{(n)}}{\mathcal{P}_j} \right\rceil \cdot C_j \\ &\quad + \left\lceil \frac{R_i''^{(n)} - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil \cdot C_k \end{aligned} \quad (3)$$

Using equations (1) and (2) it can be observed that $R_i' < R_i$ only if

$$\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil > \left\lceil \frac{R_i' - \phi}{\mathcal{P}_{trans}} \right\rceil$$

The left part of the above equation determines the number of preemptions from τ_m during the worst-case response time of τ_i, ignoring the offset information. The right side of the equation represents the number of τ_m's activations during the execution of τ_i, taking into account the activation offset ϕ. Since τ_m is periodically activated, the activation offset ϕ can reduce the interference from τ_m during the execution of τ_i with at most one activation. Hence, the above equation becomes

$$\left\lceil \frac{R_i' - \phi}{\mathcal{P}_{trans}} \right\rceil = \left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - 1 \quad (4)$$

From Equation (4) results that

$$\begin{cases} \phi \geq R_i' - \mathcal{P}_{trans} \cdot \left(\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - 1 \right) \\ \phi < R_i' - \mathcal{P}_{trans} \cdot \left(\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - 2 \right) \end{cases} \quad (5)$$

Similarly, from equations (1) and (3) can be deduced that $R_i'' < R_i$ only if

$$\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil > \left\lceil \frac{R_i'' - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil$$

Due to the periodic activation behavior of τ_k, we conclude that

$$\left\lceil \frac{R_i'' - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil = \left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - 1 \quad (6)$$

Equation (6) is valid if

$$\begin{cases} \phi & \leq & \mathcal{P}_{trans} \cdot \left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - R_i'' \\ \phi & > & \mathcal{P}_{trans} \cdot \left(\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - 1 \right) - R_i'' \end{cases} \quad (7)$$

The values of R_i' and R_i'' can be calculated using equations (2) and (3). The number of preemptions from tasks τ_m and τ_k are determined using equations (4) and (6). R_i' and R_i'' are introduced in equations (5) and (7) to determine the intervals for which these statements are valid. Task τ_i experiences its minimum worst-case response time for those values of ϕ representing the intersection of the solutions of (5) and (7). If the intersection of the solution is a void interval then τ_i's worst-case response time has no anomalous behavior with respect to variations of ϕ.

If the analyzed tasks have arbitrary deadlines, equations (2) and (3) are replaced by the equations defining the longest *level-i busy window* [7]:

$$w_i'^{(n+1)}(q) = q \cdot C_i + \sum_{j \in HP(i)}^{j \neq m} \left\lceil \frac{w_i'^{(n)}(q)}{\mathcal{P}_j} \right\rceil \cdot C_j + \left\lceil \frac{w_i'^{(n)}(q) - \phi}{\mathcal{P}_{trans}} \right\rceil \cdot C_m; w_i'^{(0)} = q \cdot C_i \quad (8)$$

and

$$w_i''^{(n+1)}(q) = q \cdot C_i + \sum_{j \in HP(i)}^{j \neq k} \left\lceil \frac{w_i''^{(n)}(q)}{\mathcal{P}_j} \right\rceil \cdot C_j + \left\lceil \frac{w_i''^{(n)}(q) - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil \cdot C_k \quad (9)$$

Furthermore, for tasks with activation jitters, equations (8) and (9) are slightly changing:

$$w_i'^{(n+1)}(q) = q \cdot C_i + \sum_{j \in HP(i)}^{j \neq m} \left\lceil \frac{w_i'^{(n)}(q) + \mathcal{J}_j}{\mathcal{P}_j} \right\rceil \cdot C_j + \left\lceil \frac{w_i'^{(n)}(q) - \phi}{\mathcal{P}_{trans}} \right\rceil \cdot C_m; w_i'^{(0)} = q \cdot C_i \quad (10)$$

and

$$w_i''^{(n+1)}(q) = q \cdot C_i + \sum_{j \in HP(i)}^{j \neq k} \left\lceil \frac{w_i''^{(n)}(q) + \mathcal{J}_j}{\mathcal{P}_j} \right\rceil \cdot C_j + \left\lceil \frac{w_i''^{(n)}(q) + \mathcal{J}_k - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil \cdot C_k \quad (11)$$

where $\mathcal{J}_i$ is the maximum activation jitter of task τ_i.

The number of preemptions from task τ_m during the busy period w_i' – Equation (10) – is not affected by the input jitter of task τ_m. If the critical instant of τ_i is determined by the simultaneous activation of τ_i and τ_k, the offset analyses presented in [15, 9, 10] assume that task τ_m is always released as soon as possible, that means τ_m is activated purely periodic. Quite the opposite, if the critical instant of τ_i – Equation (11) – is determined by the simultaneous activation of τ_i and τ_m, then the first activation of each higher priority task arrives as late as possible, while the next requests are released as soon as possible. Thus, the time span between the first and the second activation is equal to $\mathcal{P}_j - \mathcal{J}_j$.

4.2. Tasks with Correlated Activations

Assume the same task set described in Section 4.1. Ignoring the dependencies between tasks, the worst-case response time of task τ_i is calculated using Equation (1), where $HP(i)$ contains the indices of all higher priority tasks executed on the same resource as τ_i. Suppose that exists a higher priority task τ_k, $k \in HP(i)$ such that τ_k and τ_i belong to the same transaction. Obviously, τ_k and τ_i have the same activation period, labeled $\mathcal{P}_{trans}$. ϕ denotes the minimum offset between any two consecutive activations of τ_k and τ_i. Considering that the activation of τ_i is delayed with ϕ, the worst-case response time of τ_i is computed using the following equation:

$$R_i'^{(n+1)} = C_i + \sum_{j \in HP(i)}^{j \neq k} \left\lceil \frac{R_i'^{(n)}}{\mathcal{P}_j} \right\rceil \cdot C_j + \left\lceil \frac{R_i'^{(n)} - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil \cdot C_k \quad (12)$$

Since τ_k is released periodically, activating τ_i with the offset ϕ relative to τ_k's activation is equivalent with activating τ_i exactly $\mathcal{P}_{trans} - \phi$ before the next activation of τ_k. Comparing equations (1) and (12) we observe that $R_i' < R_i$ only if

$$\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil > \left\lceil \frac{R_i' - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil$$

As explained in Section 3, depending on the value of ϕ, three different behaviors of the worst-case response time of τ_i can be distinguished.

Due to the functional dependency between τ_k and τ_i, τ_i is always released after τ_k has completed its execution. The entire activation pattern of τ_k is shifted with a time interval ϕ before the activation of τ_i. If τ_i completes its execution before any additional activation of τ_k is released then, by

left-shifting the activation pattern of τ_k with ϕ may reduce the number of preemptions from τ_k by one. Since τ_k is periodically activated a further shifting of its activation pattern can not decrease the number of preemptions anymore. This can be observed comparing Figures 4(a) and 4(b). Hence, the above equation becomes

$$\left\lceil \frac{R'_i - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil = \left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - 1 \tag{13}$$

Equation (13) is valid if

$$\begin{cases} \phi & \leq & \mathcal{P}_{trans} \cdot \left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - R'_i \\ \phi & > & \mathcal{P}_{trans} \cdot \left(\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil - 1 \right) - R'_i \end{cases} \tag{14}$$

R'_i is calculated replacing the number of activations of τ_k in (12) with the right term of Equation (13). R'_i is introduced in (14) to calculate the values of ϕ for which these equations are valid. The solution of (14) represents the values of ϕ for which τ_i experiences its minimum worst-case execution time. This values correspond to the scheduling scenario presented in Figure 4(a), and bounds the first interval in Figure 5.

For values of ϕ larger than the upper bound of the previous interval, task τ_i is preempted by an additional activation of τ_k. The number of activations of τ_k during the response time of τ_i is equal to the number of preemptions described by Equation (1), i.e. $\left\lceil \frac{R_i}{\mathcal{P}_{trans}} \right\rceil$. Task τ_i experiences its longest worst-case response time.

The third behavioral interval occurs for values of ϕ such that τ_i is released together with the next activation of τ_k, or later. Since τ_k is periodically released, the interval is determined by values of ϕ for which the next equation holds:

$$\phi > \mathcal{P}_{trans} \tag{15}$$

The later the activation of τ_i, the smaller is the contribution of the first activation of τ_k at the worst-case response time of τ_i.

For tasks with arbitrary deadlines, Equation 12 is substituted by the equation defining the longest *level-i busy window* [7]:

$$\begin{aligned} w_i'^{(n+1)}(q) &= q \cdot C_i + \sum_{j \in HP(i)}^{j \neq k} \left\lceil \frac{w_i'^{(n)}(q)}{\mathcal{P}_j} \right\rceil \cdot C_j \\ &\quad + \left\lceil \frac{w_i'^{(n)}(q) - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil \cdot C_k \end{aligned} \tag{16}$$

The limits of the three intervals are determined replacing R_i by $w_i(q)$ and R'_i by $w'_i(q)$ in equations (13) and (14).

If the tasks have a periodic with jitter activation model, Equation (16) is replaced by

$$\begin{aligned} w_i'^{(n+1)}(q) &= q \cdot C_i + \sum_{j \in HP(i)}^{j \neq k} \left\lceil \frac{w_i'^{(n)}(q) + \mathcal{J}_j}{\mathcal{P}_j} \right\rceil \cdot C_j \\ &\quad + \left\lceil \frac{w_i'^{(n)}(q) + \mathcal{J}_k - (\mathcal{P}_{trans} - \phi)}{\mathcal{P}_{trans}} \right\rceil \cdot C_k \end{aligned} \tag{17}$$

Moreover, Equation (15) becomes

$$\phi > \mathcal{P}_{trans} - \mathcal{J}_k \tag{18}$$

5. Task-Pairs with Anomalous Dependencies

In the previous sections we showed that tasks with correlated release times may easily change the worst-case scheduling scenario of other tasks in the system. We carried out an analysis to detect the bounds of the intervals where the worst-case response times may have anomalous behavior. In this section we propose an algorithm to find task-pairs that may suffer from such anomalies.

Algorithm 1 Task pairs with anomalous dependencies

INPUT: The system graph $\mathcal{G} = (\mathcal{T}, \mathcal{F})$; $\mathcal{T}$, the set of tasks in the system; $\mathcal{F}$, the set of inter-tasks functional dependencies; τ_i, the task to be analyzed.

OUTPUT: $[\mathcal{T}_{anomaly}]$, the set of tasks that may generate anomalies during the execution of τ_i.

```
1: get CPU, the resource on which τ_i is executed;
2: get T_CPU, the set of tasks mapped on CPU;
3: for all τ_j ∈ T_CPU do
4:    if prio(τ_j) ≥ prio(τ_i) then
5:       insert j into HP(i);
6: get PATH, the set of paths in G = (T, F);
7: T_anomaly = ∅;
8: for all path ∈ PATH do
9:    get T_path, the set of tasks in path
10:   I = ∅
11:   for all k ∈ HP(i) do
12:      if τ_k ∈ T_path then
13:         add pathIndex(τ_k) to I;
14:   if |I| > 1 then
15:      for j = 0 to |I| − 2 do
16:         id_start = j; id_end = j + 1;
17:         for all τ_m ∈ T_path do
18:            if pathIndex(τ_m) ∈ (id_start; id_end) then
19:               add τ_m to T_anomaly;
20: return T_anomaly;
```

Algorithm 1 identifies the tasks that possibly influence the scheduling of τ_i leading to non-monotonic behavior of

the worst-case response time of τ_i. τ_i is executed on the processing element CPU. Basically, the algorithm consists of a systematic search applied on the entire task graph in order to identify the transactions containing at least two tasks, say τ_m and τ_k, mapped on CPU and with priorities higher than τ_i. If such a transaction exists, then the variation of the execution demand of any task located in that transaction between τ_m and τ_k can influence the scheduling of τ_i. Furthermore, these tasks possibly disturb also the scheduling of τ_k, where $prio(\tau_m) > prio(\tau_k)$.

Lines 3 to 5 identify the tasks mapped on CPU with a priority higher than τ_i. The algorithm selects a transaction (line 6) and determines all its tasks (line 9). During the next step, the algorithm finds the tasks in the transaction with priorities higher than τ_i (line 11 and 12) and insert their transaction indices in $\mathcal{I}$ (line 13). Each task in that transaction located between any two tasks with indices in $\mathcal{I}$ may generate a scheduling anomaly for τ_i. If the cardinality of $\mathcal{I}$ is bigger than 2, then the algorithm systematically analyzes all tasks located between the indices defined by any two consecutive elements of $\mathcal{I}$. The previous algorithm (line 9 to 19) is repeated until all transaction have been analyzed.

Notice that the index of task τ_i is also inserted in $HP(i)$ to guarantee that the transactions containing τ_i are also selected and analyzed. The algorithm complexity linearly depends on the number of transactions in the system.

6. Experiments

In this section we apply the analysis presented in Section 4 to the hypothetical distributed system depicted in Figure 1. The system configuration was already presented in detail at the beginning of Section 3. The initial execution demand of $\mathsf{T}_{\mathsf{I/O}}$ and the initial communication times of C_1 and C_2 are $C_{\mathsf{T}_{\mathsf{I/O}}} = 10$ and $C_{C1} = 5$ and $C_{C2} = 5$, respectively. Using the algorithm presented in Section 5 we determine the set of task-pairs with independent release times and the set of task-pairs with correlated activations, that may have anomalous dependencies. We find out that the worst-case response time of task T_3 may be influenced by the execution/communication demands of $\mathsf{T}_{\mathsf{I/O}}$, C_1 and C_2. The same tasks determine also the offset of T_2, and can influence the worst-case response time of T_2, as well.

Table 1 shows the behavioral intervals of the worst-case response time of T_3 depending on the execution/communication times of $\mathsf{T}_{\mathsf{I/O}}$, C_1 and C_2. As expected, task T_3 experiences its minimum worst-case response time in interval II. We apply a binary search algorithm [12] on intervals II and III to determine the eventual variations of the worst-case response time of T_3 generated by external scheduling perturbations.

As it can be seen in the column corresponding to the third interval, the worst-case response time of T_3 increases up to 340. Due to the large communication demand of C_1, the bus load increases and the scheduling of C_2 is disturbed. Because C_2 has not a constant response time, T_2 is activated within a jitter interval, leading to a larger worst-case response time. Figure 6 shows the behavior of T_3's response time corresponding to the values contained in Table 1. The worst-case response time of T_3 as function of the execution demand of $\mathsf{T}_{\mathsf{I/O}}$ was already presented in Figure 3.

	Behavioral intervals				
	I	II	III		
C_{C_1}	$[0;10)$	$[10;95]$	$(95;175]$	$(175;187]$	$(187;188]$
R_{T_3}	290	255	290	315	340
$C_{\mathsf{T}_{\mathsf{I/O}}}$	$[0;15)$	$[15;100]$	$(100;193]$		
R_{T_3}	290	255	290		
C_{C_2}	$[0;10)$	$[10;95]$	$(95;188]$		
R_{T_3}	290	255	290		

Table 1. Behavioral intervals for WCRT of T_3

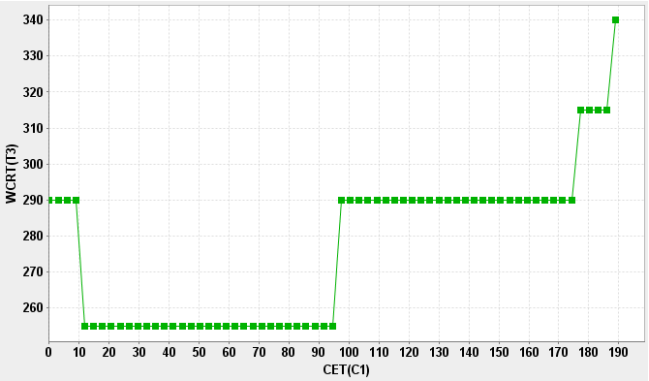
Table 2 shows the behavioral intervals of the worst-case response time of T_2 computed using the algorithm presented in Section 4.2. T_2 experiences its minimum worst-case response time in intervals I and III. We apply the same binary search algorithm to find out if the scheduling of T_2 is disturbed by external factors. Again, for large values of C_1 the high bus load affects the scheduling of C_2. Obviously, this lead to a completion jitter at C_2's output, influencing the scheduling of T_2.

	Behavioral intervals				
	I	II	III		
C_{C_1}	$[0;110]$	$(110;155]$	$(155;175]$	$(175;180)$	$[180;187]$
R_{T_2}	35	65	$(65;45)$	$(70;65)$	65
$C_{\mathsf{T}_{\mathsf{I/O}}}$	$[0;115]$	$(115;160]$	$(160;190)$		$[190;193]$
R_{T_2}	35	65	$(65;35)$		35
C_{C_2}	$[0;110]$	$(110;155]$	$(155;185)$		$[185;188]$
R_{T_2}	35	65	$(65;35)$		35

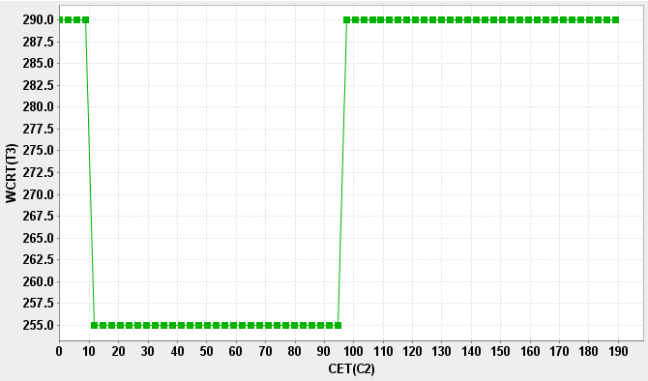
Table 2. Behavioral intervals for WCRT of T_2

Figure 7 depicts the evolution of T_2's worst-case response time as function of the communication demands of C_1 and C_2. Figure 5 at page 4 shows the behavior of T_2's worst-case response time as function of the execution demand of $\mathsf{T}_{\mathsf{I/O}}$.

Finally, we measure the global path lateness, i.e. the sum of all path latencies in the system. As expected, we identify different behavioral intervals corresponding to the behavioral intervals of the individual task response times. Figure 8 shows the evolution of the global path lateness as function of the communication demands of channels C_1 and C_2. As it can be observed, the global path lateness is piecewise linear, but it has discontinuity points exactly at the interval margins described in Section 4. Of special interest are the points where the global path lateness sinks, increas-

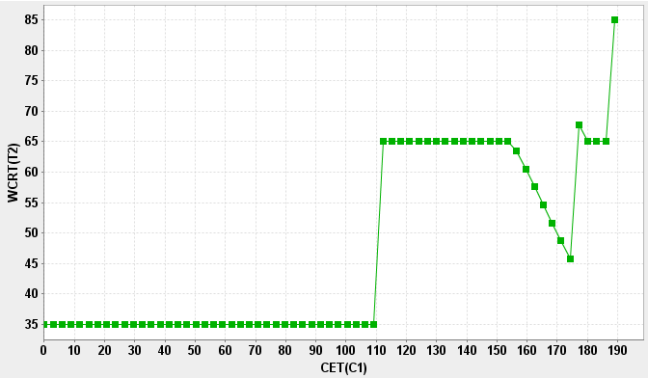


WCRT of T_3 as function of C_{C_1}

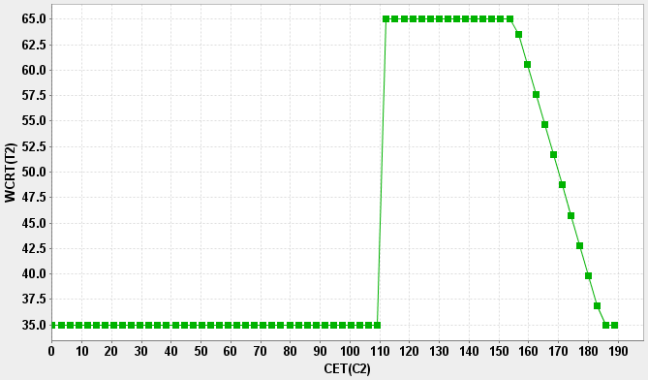


WCRT of T_3 as function of C_{C_2}

Figure 6. Task with independent release times

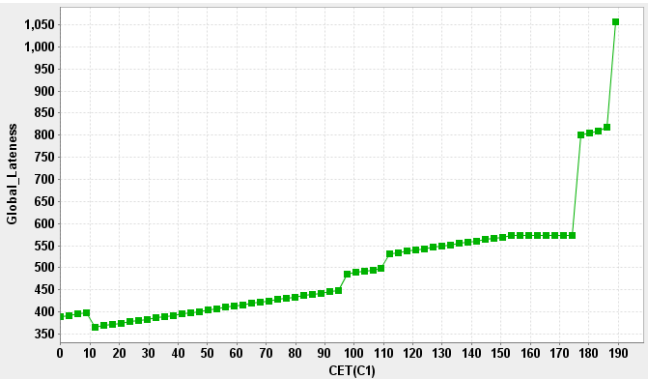


WCRT of T_2 as function of C_{C_1}

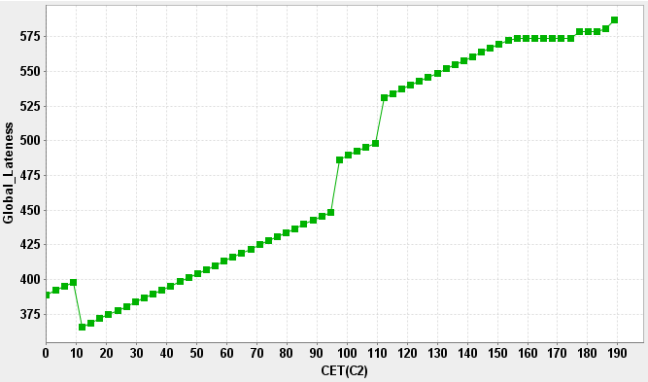


WCRT of T_2 as function of C_{C_2}

Figure 7. Tasks with correlated release time



Global path lateness as function of C_{C_1}



Global path lateness as function of C_{C_2}

Figure 8. The variation of global path lateness

ing therefore to overall system performance, even though the total load in the system has risen.

7. Conclusion

Variations of local system parameters may easily lead to unpredictable behavior of the system performance metrics. Such anomalies are very common in distributed systems with complex dependencies. As they invalidate local component worst-case assumptions, they are a potential threat to system verification and to system robustness.

In this paper, we identified the necessary and sufficient conditions for the occurrence of such anomalies and gave equations to determine their effects. This is helpful for manual design and as a support of automated design space exploration, which was shown in the examples.

Besides the contributions to analytical, formal techniques, the same methods can be used to significantly improve coverage in simulation, which is urgently demanded in industry. Component suppliers can thus document potential anomalies and provide their customers with appropriate simulation patterns – a large improvement towards robustness and optimization in todays design practice.

References

[1] S. Chakraborty. *System-Level Timing Analysis and Scheduling for Embedded Packet Processors*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2003.

[2] J. P. Gutierrez, J. J. G. Garcia, and M. G. Harbour. Best-case analysis for improving the worst-case schedulability test for distributed hard realtime systems. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS)*, 1998.

[3] M. G. Harbour and J. C. P. Gutiérrez. Response time analysis for tasks scheduled under EDF within fixed priorities. In *Proceedings of the Real-Time Systems Symposium (RTSS)*, pages 200–209, 2003.

[4] M. Jersak. *Compositional Performance Analysis for Complex Embedded Applications*. PhD thesis, Technical University of Braunschweig, 2004.

[5] M. Jersak, R. Henia, and R. Ernst. Context-aware performance analysis for efficient embedded system design. In *Proc. of Design, Automation and Test in Europe (DATE'04)*, Paris, France, Mar. 2004.

[6] H. Kopetz and G. Gruensteidl. TTP - a time-triggered protocol for fault-tolerant computing. In *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing*, pages 524–532, 1993.

[7] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the Real-Time Systems Symposium*, pages 201–209, 1990.

[8] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[9] J. C. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings of 19th IEEE Real-Time Systems Symposium (RTSS)*, Madrid, Spain, 1998.

[10] J. C. Palencia and M. G. Harbour. Exploiting precedence relations in the schedulablilty analysis of distributed real-time systems. In *Proceedings of the 20th Real-Time Systems Symposium (RTSS)*, 1999.

[11] P. Pop. *Analysis and Synthesis of Communication-Intensive Heterogeneous Real-Time Systems*. PhD thesis, Linkping University, 2003.

[12] R. Racu, M. Jersak, and R. Ernst. Applying sensitivity analysis in real-time distributed systems. In *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Francisco, CA, USA, 2005.

[13] O. Redell and M. Sanfridson. Exact best-case response time analysis of fixed priority scheduled tasks. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS)*, pages 165–172, 2002.

[14] K. Richter and R. Ernst. Event model interfaces for heterogeneous system analysis. In *Proc. of Design, Automation and Test in Europe Conference (DATE'02)*, Paris, France, Mar. 2002.

[15] K. Tindell. Adding time-offsets to schedulability analysis. Technical Report YCS 221, Department of Computer Science, University of York, UK, 1994.

[16] K. Tindell and J. Clark. Holistic schedulability analysis for distributed real-time systems. *Microprocessing and Microprogramming - Euromicro Journal (Special Issue on Parallel Embedded Real-Time Systems)*, 40:117–134, 1994.