Improved Response Time Analysis of Tasks Scheduled under Preemptive Round-Robin

Razvan Racu, Li Li, Rafik Henia, Arne Hamann, Rolf Ernst
Institute of Computer and Communication Network Engineering
Technical University of Braunschweig, Germany
{racu|lili|henia|hamann|ernst}@ida.ing.tu-bs.de

ABSTRACT

Round-Robin scheduling is the most popular time triggered scheduling policy, and has been widely used in communication networks for the last decades. It is an efficient scheduling technique for integration of unrelated system parts, but the worst-case timing depends on the system properties in a very complex way. The existing works on response time analysis of task scheduled under Round-Robin determine very pessimistic response time bounds, without considering in detail the interactions between tasks. This may lead to a degradation of the efficiency of Round-Robin scheduling algorithm, and becomes a practical obstacle to its application in real-time systems. In this paper we present an approach to compute much tighter best-case and worst-case response time bounds of tasks scheduled under preemptive Round-Robin, including also the effects of the scheduling algorithm.

Categories and Subject Descriptors

B.8 [**Performance and Reliability**]: Performance Analysis and Design Aids; C.3 [**Special-Purpose and Application-Based Systems**]: Real-Time and Embedded Systems

General Terms

Design, Performance, Reliability, Verification

1. INTRODUCTION

Round-Robin scheduling is known as a very effective time triggered scheduling policy. It allows an easy balance of resource utilization between tasks that are ready to be executed. An example is the POSIX standard. In real-time systems Round-Robin scheduling can be used as a best effort strategy. Moreover, in high load situations, it still guarantees a minimum service rate for each task, as a minimum performance guarantee. This makes it a good strategy for systems integration. For instance, *weighted Round-Robin* is the preferred scheduling strategy used in packet-based communication systems [1]. Recently, such systems has been used together with different real-time applications [4], which requires to guarantee the satisfiability of the imposed constraints.

If the deadlines in real-time systems are firm or hard, worst-case load situations must be assumed. This usually means that only the time slot assigned to a task is counted, resulting in the same performance as static time-driven scheduling, i.e. TDMA. In practice, we often find that TDMA is preferred in such cases due to its simpler control and predictable behavior. Examples are the automotive FlexRay [2] or the TTP protocols [8].

However, the pessimistic assumption is unnecessary if task properties can be taken into account, leading to a much better proven worst-case timing compared to TDMA. The corresponding schedulability analysis is, however, far more complicated. This is due to the dynamic assignment of the time slots, that depends on both the time slot order, and the activation and execution times of all other tasks sharing the resource.

There is a host of work on efficient and fair implementations of Round-Robin schedulers, most of them for communication networks [13, 16, 5, 7]. However, these works do not consider the real-time aspects of the scheduling, and if they do so, they only determine very conservative latency bounds.

Later on, several network analysis methods provide more insight into the real-time performance of networks. Some of them use probabilistic models to describe the workload arrival or the communication times of the connections [18, 20, 21], which is not adequate for the analysis of hard real-time systems.

Raha *et al.* [14] use a deterministic model for traffic description, but the computed response time bounds are still very conservative, since the analysis does not capture in detail the interaction between the communications sharing the same resource.

Migge *et al.* [12] use a mathematical model based on trajectories [11] to compute response time bounds for deterministic task models scheduled under Round-Robin within fixed-priorities.

In this paper, we present an approach to preemptive Round-Robin scheduling analysis for a set of tasks characterized by arrival functions, including burst behavior. The approach uses an event model interface to allow the analysis of tasks with deterministic workload bounds.

The rest of the paper is structured as follows: Section 2 presents the computational model with details about the workload arrival model and about the Round-Robin scheduler. Section 3 describes the worst-case execution scenario of a task and presents the algorithm to compute the worst-case response time. The algorithm flow is explained using a simple task set example. Thereafter, Section 4 presents a best-case response time algorithm based on a conservative approximation of the scenario leading to the shortest task response time. Section 5 considers the effects of the scheduling algorithm on the worst-case response time.

2. COMPUTATIONAL MODEL

2.1 Task set

In this paper we address the problem of schedulability analysis for a set of n tasks $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_n\}$, executed on a processing resource using Round-Robin arbitration strategy. The tasks are fully preemptive, i.e. their execution can be interrupted at any point in time. The tasks are assumed to be independent, i.e. no blocking occurs between tasks, and no correlation exists between the release patterns of the tasks.

CODES+ISSS'07, September 30–October 3, 2007, Salzburg, Austria.

The tasks are activated by activating events, generated in a multitude of ways, including expiration of a timer, external or internal interrupt, availability of data for processing, etc. The timing behavior of the activating events is described in detail in Section 2.2.

At each activation a task requires the resource for a bounded period of time, called task *execution time*. The minimum bound of the execution time of a task τ_i is called *best-case execution time*, denoted C_i^{min}. Analogously, the maximum bound is called *worst-case execution time* and is referred to as C_i^{max}.

Successive task activations are allowed to overlap each other. In order to prove the schedulability of the task set, one must compute the response times of the tasks and compare them against a set of predefined deadlines. There is no restriction with respect to task deadlines. This assumption shows the generality of the response time analysis presented in this paper.

2.2 Event models

The activating events of a task are captured by an *event stream*. The behavior of an event streams is described using *event models*.

The classical event models used in the scheduling theory use a set of parameters to describe the timing of the events within an event stream. Hence, a *periodic event model* is characterized by the *period* ($\mathcal{P}$) [9], a *periodic with jitter event model* [19] is described by two parameters, the *period* ($\mathcal{P}$) and the *jitter* ($\mathcal{J}$) and a *sporadic event model* [17] is described by the *minimum inter-arrival distance* (d^-).

For the purpose of the response time analysis, the event models are mathematically represented using a set of functions:$\eta^+(\Delta t)$, $\eta^-(\Delta t)$, $\delta^-(n)$ and $\delta^+(n)$ [15]. The η functions are called arrival functions and describe the maximum and minimum number of events arriving in the interval Δt. The $\delta(n)$ functions are called $distancefunctions$ and describe the minimum and maximum distance between any event i and the $(n-1)$-th event after i in the corresponding event stream [15].

In this paper, we use the four functions to capture the workload of the tasks in the response time equations.

2.3 Round-Robin scheduler

The Round-Robin (RR) scheduler assigns the resource to each task in a cyclic fashion. Each task τ_i has assigned a predefined amount of time, called *time slot*, where it is granted the resource. For brevity, we denote the time slot of τ_i by θ_i. The sum of all time slots is called *RR-turn*. The order in which the time slots are executed within a RR-turn is predefined. We assume the time slots are ordered according to the corresponding task indices. After θ_n a new RR-turn starts. During each turn every task accesses the resource only within its time slot.

If no instance of τ_i is pending at the beginning of θ_i, then no processor time is assigned to task τ_i and the scheduler examines τ_{i+1}. If at least one activation of τ_i is waiting for execution at the beginning of θ_i, then τ_i is allowed to execute without interruption for at most θ_i units of time. If the task did not complete its execution at the end of its time slot, it is interrupted (preempted) by the scheduler. In case that τ_i completes its execution before its time slot expires, the next pending activations of τ_i, are executed for the time left in the current time slot. If the time slot did not expire and there are no more waiting task instances, the scheduler switches to the task corresponding to the next slot in the RR-turn.

3. WORST-CASE RESPONSE TIME

3.1 Critical instant

In order to find the worst-case response time, one must determine the scheduling scenario when the longest response time occurs. Liu and Layland [10] defined the *critical instant* of a task to be the instant at which a request for that task will have the largest response time. Lehoczky [9] proved that the critical instant of a task τ_i on a fixed-priority scheduler, occurs at the beginning of the *level-i busy period*. A *level-i busy period* is defined as the maximum time for which a processor executes tasks with priority higher than or equal to the priority of τ_i. However, under Round-Robin, the task priorities dynamically change in time, and therefore, the definition of the level-i busy period must be first adapted and then applied for the computation of the worst-case response time.

In general, the *maximum busy period* is the maximum time for which a processor is busy executing tasks. Intuitively, the critical instant of a task under Round-Robin is the instant when the task has the lowest priority among all other tasks. This scenario guarantees the *longest waiting time* until the task may access the resource again.

LEMMA 1 (MAXIMUM BUSY PERIOD). *The maximum busy period occurs when all tasks mapped on the resource are simultaneously released and request the worst-case execution time. Additionally, all subsequent task activations arrive as soon as possible (worst-case load condition).*

PROOF. As in [10, 6, 9]. □

THEOREM 1 (CRITICAL INSTANT UNDER ROUND-ROBIN). *The critical instant of a task under Round-Robin occurs at the beginning of the* maximum busy period *instantiated right after the time slot of the analyzed task expired.*

PROOF. Let τ_n be the analyzed task and t_0 the time instant when the time slot θ_n expires. The beginning of the next θ_n occurs at:

$$t = t_0 + \sum_{j=1}^{n-1} \min(\theta_j, w_j) \tag{1}$$

where w_j represents the maximum execution time requested by task τ_j up to the start time of its slot, denoted as $t_{(j)}$. Using the standard event model interface described in Section 2.2, w_j can be calculated as

$$w_j = C_j^{max} \cdot \eta_j^+(t_{(j)}) \text{ where } t_{(j)} = \sum_{k=1}^{j-1} w_k \tag{2}$$

Assume that τ_n is released with an offset ϕ_n after t_0. Since τ_n can not start the execution earlier than t, the response time of τ_n will be ϕ_n units of time shorter. Obviously, the longest response time occurs for $\phi_n = 0$.

If τ_n is released before t_0, then τ_n gets the chance to start its execution in the slot finishing at t_0, and thus, its waiting time may be shorter than the case when τ_n starts at t_0. Hence, t_0 represents the critical instant. □

3.2 Worst-case response time algorithm

Consider a task set consisting of four preemptive tasks, $T1$, $T2$, $T3$ and $T4$ executed on a Round-Robin resource. $T1$, $T2$, $T3$ are periodically activated. Task $T4$ experiences a burst activation due to a large jitter. The distance between the burst events is controlled by the minimum distance parameter (d^-) of the input event model. After the burst events expired, the task is released periodically, as it can be seen in Figure 1.

Table 1 shows the parameters of the input event models, the worst-case execution times, and the time slot values corresponding to the task set. The time slots are scheduled in the following order: $\theta_1 \rightarrow \theta_2 \rightarrow \theta_3 \rightarrow \theta_4$.

Table 1: The task parameters

Task	Activation model			Execution time	Time slot
(τ_i)	$\mathcal{P}_i$	$\mathcal{J}_i$	d^-	(C_i^{max})	θ_i
$T1$	15	-	-	3	2
$T2$	50	-	-	10	3
$T3$	30	-	-	7	5
$T4$	20	50	5	5	7

The algorithm to determine the worst-case response time of a task under Round-Robin uses the *windowing technique* proposed

by Lehoczky [9], and refined latter by Tindell [19]. The approach finds the worst-case response time of a task by checking different *time windows* within the *busy period*. The *q-window* of task τ_i, denoted also $w_i(q)$, represents the time interval between the instant when the *maximum busy period* was initiated (t_0) and the time instant when the q-th activation of t_i completes its execution. In other words, the *q-window* of τ_i represents the response time of the first q activations of τ_i in the *maximum busy period*.

Algorithm 1 Worst-case response time under Round-Robin

INPUT: $C_k^{max}, \forall k \leq n$, the worst-case execution time of τ_k;
$\theta_k, \forall k \leq n$, the predefined time slot of τ_k;
$\mathcal{EM}_k, \forall k \leq n$, the activation model of τ_k
OUTPUT: R_i^{max}, the worst-case response time of τ_i

1: $q = 0$;
2: $R_i^{max} = 0$;
3: **repeat**
4: $q++$;
5: $w_i(q) = q \cdot C_i^{max} + \mathcal{I}(q)$;
6: $R_i(q) = w_i(q) - \delta_i^-(q)$;
7: $R_i^{max} = max(R_i^{max}, R_i(q))$;
8: **until** $\eta_i^+(w_i(q)) = q$

Algorithm 1 computes the worst-case response time of τ_i. During each loop iteration the *q-th window* of τ_i is analyzed. The value of $w_i(q)$ (line 5) is equal to the execution demand of the first q activations of τ_i, $q \cdot C_i$, plus the interference, $\mathcal{I}(q)$, with other tasks mapped on the same resource. $\mathcal{I}(q)$ is explained in detail later in this section.

$\delta_i^-(q)$ returns the arrival time of the q-th activation of τ_i. Line 7 returns the response time of the q-th activation of τ_i within the busy window. The maximum over all $R_i(q)$ represents the worst-case response time R_i^{max} of τ_i.

The loop terminates when the number of activations released in the *q-th window* does not exceed q, i.e. the first q activations of τ_i have been completely processed before the $(q+1)$-th activation arrived. The maximum number of activations that arrive in the time window $w_i(q)$ is computed using the *upper-bound arrival function*, η_i^+.

Notice that both functions, δ_i^- and η_i^+ are obtained from the activation model $\mathcal{EM}_i$ of τ_i.

Figure 1 shows the worst-case scheduling scenario of task $T4$ in the example introduced above. The worst-case response time algorithm starts with $q = 1$ and calculates the value of the 1st window of $T4$. As we can observe in the figure, $w_4(1) = 15$, and, since first activation of $T4$ was released at $t_0 = 0$, the response time of this activation is also 15. Moreover, we observe that the 2nd activation of $T4$ was released within the 1st window, and thus, the algorithm investigates also the 2nd window. The value of $w_4(2)$ is equal to 27 and, because the 2nd activation of $T4$ was released at time instant 5, its response time is 22. The algorithm continues until $q = 7$. We observe that the 8th activation of $T4$ is released after the 7th activation has completed the execution, and therefore the algorithm terminates. The worst-case response time of $T4$ occurs for the 3rd and 4th activations of the task.

The interference $\mathcal{I}(q)$ of the analyzed task with other tasks, during the *q-th window* is computed as:

$$\mathcal{I}(q) = \sum_{k=1}^{K(q)} \mathcal{I}_k, \text{ where } K(q) = \left\lceil \frac{q \cdot C_i^{max}}{\theta_i} \right\rceil \quad (3)$$

where $\mathcal{I}_k$ represents the interference with other tasks during the k-th RR-turn, and $K(q)$ is the total number of RR-turns required by q activations of τ_i to complete. In other words, $\mathcal{I}_k$ represents the total time that all other tasks used the processor during turn k.

For example, in Figure 1, the time interval $[31, 41]$ corresponds to $\mathcal{I}_3$, and represents the time tasks $T1, T2$ and $T3$ used the processor in the third turn. Every $\mathcal{I}_k$ is defined as $\mathcal{I}_k = \sum_{j=1}^{n-1} \mathcal{I}_{k,j}$, where $\mathcal{I}_{k,j}$ is the interference in the k-th RR-turn with the task whose predefined time slot order in the RR-turn is equal to j, assuming that the time slot of the analyzed task has the predefined time slot order n.

If we refer again to Figure 1, the time interval $[22, 24]$ corresponds to $\mathcal{I}_{2,3}$ and represents the time spent by $T3$ in the second turn.

The interference $\mathcal{I}_{k,j}$ is determined by the remaining execution demand of τ_j at the beginning of θ_j in the k-th turn.

Since the remaining demand might not necessarily use up the entire time slot θ_j, one must iteratively check if new activations of τ_j that arrived during the consumed time of θ_j, can be processed in the available time in θ_j. Therefore, $\mathcal{I}_{k,j}$ is described as follows:

$$\mathcal{I}_{k,j} = \sum_{r=1}^{R} pad_{k,j}(r), R \text{ such that } pad_{k,j}(r) \neq 0 \quad (4)$$

A *pad* represents the time spent in a time slot by all unserved task activations already arrived before the start time of the pad.

For example, if a slot contains multiple pads, the first pad in the slot corresponds to the execution of the task activation arrived before the slot started and not served in the previous slots. The value of the second pad corresponds to the execution of the task activations arrived during the execution of the first pad, and so on. It is very important to notice that the value of a pad can not be larger than the unused time in the corresponding slot.

The value of R in Equation 4 defines the maximum number of pads in a slot. The pads in a slot are indexed over r. $pad_{k,j}(r)$ represents the r-th pad in time slot θ_j of the k-th turn.

In Figure 1, the only slot with multiple pads is the slot of $T2$ in the fourth RR-turn: one is used to finish the first activation of $T2$ and the other one is used to execute a part of the second activation of $T2$, which arrived during de execution of $pad_{4,2}(1)$.

The value of a pad in time slot θ_j is given by the remaining execution demand of τ_j at the beginning of the pad and the free amount of time in θ_j.

$$pad_{k,j}(r) = \min(l_{k,j}(r), C_j^{max} \cdot \eta_j^+(t_{k,j} + \sum_{s=1}^{r-1} pad_{k,j}(s)) - \sum_{p=1}^{k-1} \mathcal{I}_{p,j} - \sum_{s=1}^{r-1} pad_{k,j}(s)) \quad (5)$$

The first term in the *min* function in Equation 5 represents the unused time in time slot θ_j at the beginning of $pad_{k,j}(r)$. This time is calculated as follows:

$$l_{k,j}(r) = \theta_j - \sum_{s=1}^{r-1} pad_{k,j}(s) \quad (6)$$

The second term in the *min* function represents the remaining execution demand of τ_j at the beginning of $pad_{k,j}(r)$. η_j^+ returns the total number of activations of τ_j since t_0. $t_{k,j}$ represents the starting time of θ_j relative to t_0, in the k-th RR-turn.

$$t_{k,j} = \sum_{p=1}^{k-1} \mathcal{I}_p + (k-1)\theta_i + \sum_{u=1}^{j-1} \mathcal{I}_{k,u} \quad (7)$$

The sum of the first and second terms in Equation 7 represents the value of the first $(k-1)$ RR-turns. The last term represents the time consumed in the k-th RR-turn by tasks with time slot order higher than τ_j.

In Figure 1, the value of $t_{5,3}$ is calculated as the interference in the first four RR-turns, plus the time $T4$ has been executed in the first four turns, plus the execution of $T1$ and $T2$ in the fifth turn.

The value of $pad_{k,j}(r)$ in Equation 5 depends on the values of

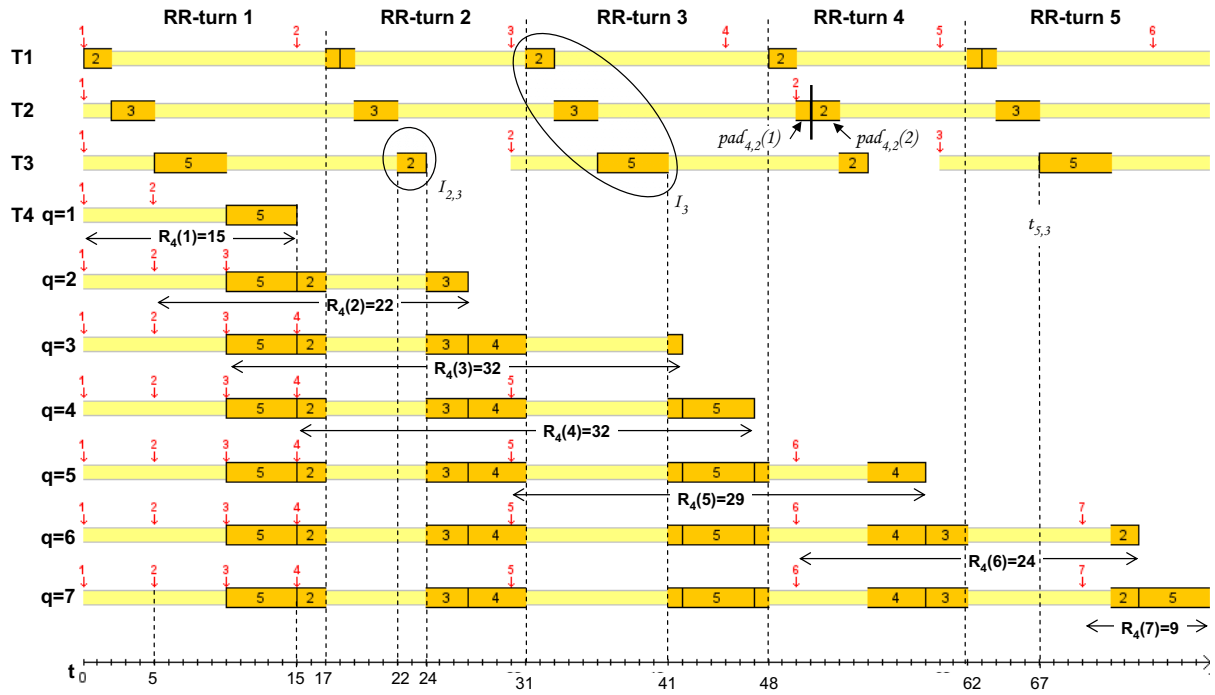

Figure 1: Worst-case response time scenario for $T4$

all $pad_{k,j}(s)$, where $s < r$. $pad_{k,j}(1)$ is calculated as follows:

$$pad_{k,j}(1) = \begin{cases} \theta_j & E_{k,j} \geq \theta_j \\ E_{k,j} & E_{k,j} < \theta_j \end{cases} \quad (8)$$

$E_{k,j}$ in Equation 8 represents the remaining execution demand of task τ_j at the beginning of θ_j, in the k-th RR-turn.

$$E_{k,j} = C_j^{max} \cdot \eta_j^+(t_{k,j}) - \sum_{p=1}^{k-1} \mathcal{I}_{p,j} \quad (9)$$

The values of $pad_{k,j}$ belonging to θ_j are iteratively calculated until first $pad_{j,k}$ value equal to zero is found. In Equation 5 this means that either the time slot θ_j has been entirely used up, or the execution demand of τ_j has been completely processed. The value of R in Equation 4 represents the total number of *pads* iteratively calculated using Equation 5.

Table 2 shows the worst-case response times of the task set, computed using the Algorithm 1 and the algorithms proposed by Raha [14] and Migge [12]. Tasks $T1$ and $T2$ are unschedulable according to Raha's algorithm. This is because the load of each task is larger than the service rate assigned to these tasks. Notice that, the response times computed by Algorithm 1 and the approach proposed in [12] are identic. However, in our opinion, the analysis in [12] suffers from the complex computational model.

Table 2: Computed worst-case response times

	Algorithm 1	Raha [14]	Migge [12]
$T1$	46	unschedulable	46
$T2$	60	unschedulable	60
$T3$	31	31	31
$T4$	32	35	32

4. BEST-CASE RESPONSE TIME

4.1 The least critical instant

The best-case response time analysis computes the shortest response time of a task. Intuitively, in order to experience the shortest response time, the task must be released at the instant when it has the highest priority among all other tasks. This occurs whenever the task is released at the beginning of its time slot.

Opposite to the concept of *maximum busy period*, one can define the *maximum idle period* with respect to a given set of tasks, as the maximum time for which the resource does not process any task in the given set.

THEOREM 2 (BEST-CASE RESPONSE TIME). *The best-case response time of a task τ_i under Round-Robin scheduling occurs when the task is released at the beginning of the maximum idle period of all other tasks mapped on the same resource as τ_i.*

PROOF. Let $\mathcal{G} = \mathcal{T} - \{\tau_i\}$ be the set of tasks executed on the same resource as τ_i, and let t_0 be the time instant indicating the beginning of the maximum idle period of all tasks in $\mathcal{G}$. If τ_i is released before t_0, then at least one task in $\mathcal{G}$ is active and interferes with τ_i.

If τ_i is released after t_0, then the interference of τ_i with the next activations of the tasks in $\mathcal{G}$ occurs earlier, and may extend the response time of τ_i. □

Contrary to maximum busy period, the *maximum idle period* makes assumptions about the completion of the tasks and not about task activation. Moreover, the dynamic behavior of the priority function in Round-Robin scheduling makes difficult to find out a relation between the task activation scenario leading to the maximum idle period.

Instead, we use a hypothetical scenario similar to [3] that leads to a conservative *maximum idle time*. We use this scenario to construct the *least critical instant* of a task τ_i scheduled under Round-Robin:

1. Task τ_i is released at an instant t_0.
2. Each task $\tau_j, j \neq i$ is released C_j^{min} time units before t_0, executes without interruption and completes its execution at t_0. The next activations of the tasks arrive as late as possible.

The second statement represents the *best-case load condition* and ensures that the interference of τ_i with other tasks occurs as late as possible. Obviously, such a scenario is not possible in practice, as all tasks must *use* the resource *simultaneously* before t_0, in order to finish at the same time.

The first statement guarantees that τ_i gets executed immediately after its activation, since at t_0 no other task except τ_i is waiting for execution.

Figure 2 shows the hypothetical scenario, where t_0 indicates the *least critical instant*.

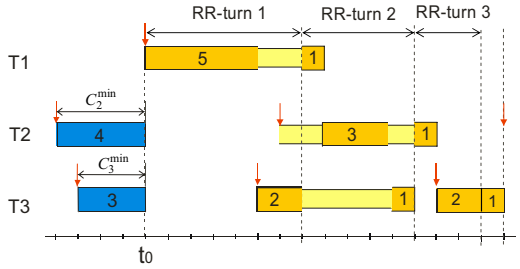

Figure 2: The least critical instant under RR

4.2 Best-case response time algorithm

In order to find the shortest response time of τ_i only the execution of the first activation of τ_i must be analyzed. The best-case response time is

$$R_i^{min} = C_i^{min} + \mathcal{I}^{min} \tag{10}$$

where C_i^{min} represents the minimum execution time of τ_i. The interference $\mathcal{I}^{min}$ of τ_i with other tasks is defined as

$$\mathcal{I}^{min} = \sum_{k=1}^{K'} \mathcal{I}_k, \text{ where } K' = \left\lceil \frac{C_i^{min}}{\theta_i} \right\rceil - 1 \tag{11}$$

Since τ_i is activated at the beginning of its time slot, there is no interference in the last RR-turn. Therefore, only the interference in the first $\lceil \frac{C_i^{min}}{\theta_i} \rceil - 1$ turns must be calculated.

The interference $\mathcal{I}_k$ in the k-th RR-turn is defined as $\mathcal{I}_k = \sum_{j=2}^{n} \mathcal{I}_{k,j}$ where $\mathcal{I}_{k,j}$ is the interference in the k-th RR-turn with the task whose predefined time slot order is equal to j, assuming that the time slot order of the analyzed task is 1.

$\mathcal{I}_{k,j}$ is computed using Equation 4. The values of $pad_{k,j}$ in Equation 4 are calculated as follows:

$$pad_{k,j}(r) = \min(l_{k,j}(r), C_j^{min} \cdot \eta_j^-(t_{k,j} + C_j^{min} + \sum_{s=1}^{r-1} pad_{k,j}(s)) - \sum_{p=1}^{k-1} \mathcal{I}_{p,j} - \sum_{s=1}^{r-1} pad_{k,j}(s))$$

The η_j^- function in above equation returns the total number of activations of task τ_j arrived until the beginning of $pad_{k,j}(r)$.

The unused time in the time slot θ_j at the end of $pad_{k,j}(r-1)$ is computed using Equation 6. The time at the beginning of θ_j in the k-th turn is

$$t_{k,j} = \sum_{p=1}^{k-1} \mathcal{I}_p + k \cdot \theta_i + \sum_{u=2}^{j-1} \mathcal{I}_{k,u} \tag{12}$$

Notice that, contrary to worst-case scenario, $t_{k,j}$ contains also the execution of τ_i in the k-th turn, as θ_i is the first time slot scheduled in the RR-turn.

The first value of $pad_{k,j}$, is calculated using Equation 8 in Section 3.2, where $E_{k,j}$ is determined by:

$$E_{k,j} = C_j^{min} \cdot \eta_j^-(t_{k,j} + C_j^{min}) - \sum_{p=1}^{k-1} \mathcal{I}_{p,j} \tag{13}$$

5. SCHEDULING OVERHEAD

The necessary schedulability condition valid for any task set and scheduling strategy is

$$U_{overall} \leq 1 \tag{14}$$

where $U_{overall}$ represents the overall resource utilization, and is defined as

$$U_{overall} = \sum_{i=1}^{n} \frac{C_i^{max}}{\mathcal{P}_i} + U_{RTOS} \tag{15}$$

In other words, the overall resource utilization is the sum of the average load generated by the tasks executed on that resource and the load generated by the scheduling algorithm.

In Round-Robin scheduling, the RTOS subroutine is executed whenever the scheduler reevaluates the task priorities, i.e when a time slot expires, or when the execution demand of the currently executed task has been completely processed. We call *scheduler task*, denoted τ_{OS}, the RTOS subroutine executed in order to reassign the task priorities.

The upper bound for the RTOS load, U_{RTOS}, is computed as

$$U_{RTOS} = \sum_{i=1}^{n} \frac{\left\lceil \frac{C_i^{max}}{\theta_i - C_{OS}} \right\rceil \cdot C_{OS}}{\mathcal{P}_i} \tag{16}$$

where C_{OS} represents the maximum execution demand of the scheduler task.

Obviously, each task τ_i requires maximum $\left\lceil \frac{C_i^{max}}{\theta_i - C_{OS}} \right\rceil$ time slots θ_i, in order to complete its execution. Notice that, in every time slot θ_i, C_{OS} units of time are used by the scheduler task. Assuming one execution of τ_{OS} at the beginning of every time slot, one can compute the RTOS overhead corresponding to the execution of each task, and therefore to determine the total RTOS overhead for the scheduling of the entire task set.

In the following we assume that no scheduling overhead occurs due to switching between different task instances executed in the same time slot. This is not a limitation of the approach, and the response time equations can be easily extended to account also for this aspect.

To capture the time spent during context switches, line 5 in Algorithm 1 is modified as follows:

$$w_i(q) = q \cdot C_i^{max} + \mathcal{I}(q) + \left\lceil \frac{q \cdot C_i^{max}}{\theta_i - C_{OS}} \right\rceil \cdot C_{OS} \tag{17}$$

The value of $K(q)$ in Equation 3 changes to:

$$K(q) = \left\lceil \frac{q \cdot C_i^{max}}{\theta_i - C_{OS}} \right\rceil \tag{18}$$

In every RR-turn there is one execution of the scheduler task at the beginning of every time slot θ_j only if the execution demand of τ_j at the beginning of θ_j is larger than zero.

The weighting function $\omega_{k,j}$ captures the execution demand of τ_j at the beginning of its time slot in the k-th RR-turn.

$$\omega_{k,j} = \begin{cases} 0 & E_{k,j} = 0 \\ 1 & E_{k,j} \neq 0 \end{cases} \tag{19}$$

where $E_{k,j}$ is defined as

$$E_{k,j} = C_j^{max} \cdot \eta_j^+(t_{k,j}) - \sum_{p=1}^{k-1} (\mathcal{I}_{p,j} - \omega_{p,j} \cdot C_{OS}) \tag{20}$$

Equation 4 changes to

$$\mathcal{I}_{k,j} = \omega_{k,j} \cdot C_{OS} + \sum_{r=1}^{R} pad_{k,j}(r) \tag{21}$$

The *upper-bound arrival function* η_j^+ in Equation 5 must also consider the task activations that eventually arrived during the execution of the *scheduler task* at the beginning of time slot θ_j. The value of $pad_{k,j}$ is computed as follows:

$$\begin{aligned} pad_{k,j}(r) &= \min(l_{k,j}(r), C_j^{max} \cdot \eta_j^+(t_{k,j} + C_{OS} + \\ &\quad \sum_{s=1}^{r-1} pad_{k,j}(s)) - \sum_{p=1}^{k-1}(\mathcal{I}_{p,j} - \omega_{p,j} \cdot C_{OS}) - \\ &\quad \sum_{s=1}^{r-1} pad_{k,j}(s)) \end{aligned} \tag{22}$$

The calculation of the unused time in time slot θ_j at the beginning of $pad_{k,j}$ changes to

$$l_{k,j}(r) = \theta_j - \omega_{k,j} \cdot C_{OS} - \sum_{s=1}^{r-1} pad_{k,j}(s) \tag{23}$$

The time instant at the beginning of θ_j in the k-th turn is computed using Equation 7.

The initial value of $pad_{k,j}$ must also account for the effects of the *scheduler task*, as follows:

$$pad_{k,j}(1) = \begin{cases} \theta_j - C_{OS} & E_{k,j} \geq \theta_j - C_{OS} \\ E_{k,j} & E_{k,j} < \theta_j - C_{OS} \end{cases} \tag{24}$$

where $E_{k,j}$ is calculated using Equation 20.

Consider again the example presented in Section 3. We carried out the worst-case response time analysis for the defined task set, this time considering the influences of the scheduler tasks. The scheduler task is assumed to have a constant execution demand $C_{OS} = 0.2$.

Table 3 shows the influence of τ_{OS} on the response times of the tasks.

Table 3: The worst-case response times incl. the OS overhead

Task	$T1$	$T2$	$T3$	$T4$
w/o τ_{OS}	46	60	31	32
w/ τ_{OS}	60	61.6	31.4	33

6. CONCLUSION

In this paper we addressed the problem of schedulability analysis of tasks under Round-Robin. First, we presented an exact algorithm to compute the worst-case response time of a task, assuming the task can be preempted at any instant. Then, we derived a best-case response time algorithm using a conservative approximation of the best-case scheduling scenario. In the last section we extended the worst-case response time equations to consider the timing overhead due to the scheduler task. The model used for computation uses a 4-function interface to describe the workload arrival. This interface easily allows the analysis of tasks with arbitrary arrival models. Table 4 shows the characteristics of different Round-Robin scheduling analyses. Notice that, the approach presented in this paper supports complex analysis scenarios, and the simple underlying model makes it very intuitive for the analysis of real-time applications.

Table 4: Characteristics of different Round-Robin scheduling analyses

	Algorithm 1	Raha [14]	Migge [12]
WCRT	√	√	√
BCRT	√	-	-
OS Overhead	√	-	-
Jitter, Burst	√	-	√
Arbitrary deadlines	√	-	-
Arbitrary arrival models	√	√	√

7. REFERENCES

[1] R. Budruk, D. Anerson, and T. Stanley. *PCI-Express System Architecture*. MindShare, Inc., 2003.

[2] FlexRay. `http://www.flexray.com/`

[3] 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.

[4] S. Heithecker, A. do Carmo Lucas, and R. Ernst. A High-End Real-Time Digital Film Processing Reconfigurable Platform. *EURASIP Journal on Embedded Systems, Special Issue on Dynamically Reconfigurable Architectures*, 2007:Article ID 85318, 15 Pages, 2007.

[5] K. Jeffay, F. D. Smith, A. Moorthy, and J. Anderson. Proportional Share Scheduling of Operating System Services for Real-Time Applications. In *IEEE Real-Time Systems Symposium*, Madrid, Spain, 1998.

[6] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, 1986.

[7] S. Kanhere and H. Sethu. On the Latency Bound of Pre-Order Deficit Round Robin. In *Proceedings of the IEEE Conference on Local Computer Networks*, pages 508–517, Tampa, FL, 2002.

[8] H. Kopetz and G. Bauer. The Time-Triggered Architecture. *Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software*, 2001.

[9] J. Lehoczky. Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. In *Proceedings of the Real-Time Systems Symposium*, pages 201–209, 1990.

[10] 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.

[11] J. Migge. *Real-time scheduling: a trajectory based model*. PhD thesis, Nice University, Nancy, France, 1999.

[12] J. Migge, A. Jean-Marie, and N. Navet. Timing analysis of compound scheduling policies: application to Posix1003.1b. *Journal of Scheduling*, 6(5):457–482, 2003.

[13] S. Mukherjee, D. Saha, M. Saksena, and S. K. Tripathi. A Bandwidth Allocation Scheme for Time Constrained Message Transmission on a Slotted Ring LAN. In *IEEE Real-Time Systems Symposium*, Raleigh-Durham, NC, 1993.

[14] A. Raha, S. Kamat, and W. Zhao. Guaranteeing End-to-End Deadlines in ATM Networks. In *International Conference on Distributed Computing Systems*, Vancouver, Canada, 1995.

[15] K. Richter. *Compositional Performance Analysis*. PhD thesis, Technical University of Braunschweig, 2004.

[16] M. Shreedhar and G. Varghese. Efficient Fair Queueing Using Deficit Round Robin. In *SIGCOMM*, pages 231–242, Boston, 1995.

[17] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic Task Scheduling for Hard Real-Time Systems. *Journal of Real-Time Systems*, 1(1):27–60, 1989.

[18] H. Takagi. Exact analysis of round-robin scheduling of services. *IBM J. Res. Dev.*, 31(4):484–488, 1987.

[19] K. Tindell, A. Burns, and A. Wellings. An Extendible Approach for Analysing Fixed Priority Hard Real-Time Systems. *Journal of Real-Time Systems*, 6(2):133–152, Mar 1994.

[20] R. Wu and Y.-B. Chen. Analysis of a loop Transmission System with Round-Robin Scheduling of Services. *IBM J. Res. Dev.*, 19(5):486–493, 1975.

[21] R. Yates. *High Speed Round Robin Queueing Networks*. PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1990.