Quality-Assuring Scheduling

Claude-J. Hamann Lars Reuther Jean Wolter Hermann Härtig
Technische Universität Dresden, Germany

Abstract

Quality-Assuring Scheduling (QAS) has been invented to provide statistical guarantees to real-time applications with variable execution times scheduled in a reservation-based fixed-priority system. The admission control is based on a probabilistic model to ensure that a requested percentage of jobs of a periodic application is successfully executed over a longer time span even in permanent overload situations.

However, the original approach is restricted to task sets with uniform periods and using nonpreemptible resources. This work overcomes these limitations. Now we extend the QAS admission algorithm to support task sets with harmonic and arbitrary periods. Our evaluation of the presented admission models shows the nearly full compliance of the predicted qualities with both simulations and measurements using a prototype real-time system.

1. Introduction

Traditional real-time admission methods use fixed, worst-case execution times (WCET) that may exceed average-case execution times by an order of magnitude for applications with varying real-time requirements. This results in low resource utilization. On the other hand, often such applications might tolerate occasional deadline misses. Several methods using statistical approaches have been developed to handle this situation. One of these approaches is *Quality-Assuring Scheduling* (QAS) [9]. We developed QAS to improve resource utilization while maintaining predictable system behavior in case of overload. QAS achieves these goals with two ideas.

- By using probabilistic distributions to describe the varying resource demands of a periodic application, the QAS admission criterion accurately models the actual scheduling in a real system: when a task does not consume its worst-case execution time, a task with a lower priority is started immediately.
- The jobs of a periodic task are split into one mandatory part and one (or more) optional part. Mandatory parts have to be executed under all circumstances. Optional parts may be aborted or dropped in case of resource shortage. QAS ensures that a minimum percentage (specified by the application) of optional parts is successfully completed over an arbitrary long period of time.

In order to achieve the requested quality of optional parts, the admission control computes the resource amount – called *reservation time* – that is required by an optional part during each period. The scheduling is based on the periodic execution of tasks, combining fixed priorities and reservations. Resource schedulers are responsible for the enforcement of the reservation times; optional parts are not allowed to consume more resources within a period than assigned by the reservation time. Resources not consumed by any task in a period can be employed to further improve the qualities of optional parts, but without giving any guarantees. Thus, we are able not only to predict, but also to control the system behavior.

We demonstrated the approach with respect to nonpreemptible resources such as disks in [9]. The admission model is given, i.e., the priority assignment and a formula to compute the reservation times. However, the model is restricted to uniform periods (all tasks have the same period length), which represents a strong limitation. With the work presented in this paper we remove this limitation by extending the admission model to task sets with harmonic periods and arbitrary periods and by including preemptible resources.

The remainder of this paper is organized as follows. The next section explaines basic ideas and general approach of QAS. After that, we will give a detailed description of priority assignment and computation of reservation times in the case of preemptible resources. Section 4 summarizes the results for nonpreemptible resources. We will validate the QAS approach by simulation experiments as well as by measurements using a prototype real-time system [11] in Section 5. Finally, a discussion of related work and a summary conclude the paper.

2. The QAS Approach

The basic ideas of QAS are

- to consider the varying execution times of jobs of a periodic task as random variables with given distributions,
- to split the jobs of such a task into one mandatory part *M* and one optional part *O* (see Fig. 1),
- to express a "quality" for each task through a requested percentage *q* of successfully completed optional parts,
- to use scheduling based on fixed priorities and reservations.

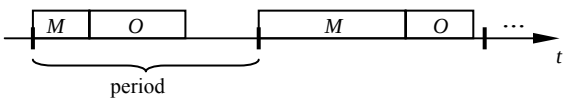


Figure 1. Periodic application consisting of mandatory and optional parts.

The reservation time r – the amount of resource to allocate to a task's optional parts during each period – is calculated based on a probabilistic model. Assuming that the scheduler aborts an optional part when it exceeds its reservation time, the reservation time r primarily results from the q-quantile of the execution time distribution (see Fig. 2(a); p.d.f.: probability density function). However, the final model has to consider that an optional part is aborted at the end of its period (the relative deadline d), even if this part has not yet exhausted its reservation time (Fig. 2(b)).

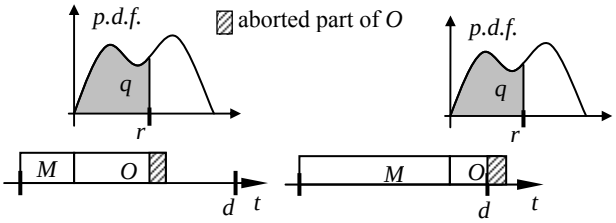


Figure 2. Abort of an optional part.

Hence, the reservation time for the optional parts O of a task requiring a quality q is the shortest time r where

$$\mathbf{P}(O \text{ does not run longer than } r \wedge O \text{ is completed until the period-end}) \geq q. \quad (1)$$

More formally, let $p_i(r)$ denote the probability that an optional part of task T_i is completed in the sense of Formula (1) ($r \in \mathbb{R}$, $i \in \mathbb{N}$). Then we obtain a system of equations for a task set $\boldsymbol{T} = \{T_1,...,T_n\}$ with requested qualities $q_1,...,q_n$:

$$r_i = \min(r \in \mathbb{R} \mid p_i(r) \geq q_i), \quad i = 1,...,n. \quad (2)$$

Since all mandatory parts have to meet their deadlines under all circumstances, the general admission criterion is

(A1) All mandatory parts must meet their deadline (which is given by the period-end).

(A2) The system of equations in Formula (2) is solvable.

Whereas [9] demonstrates the approach for nonpreemptible resources, we choose preemptible resources here in order to explain the approach in detail. In this context, preemptible means that the scheduler suspends a job when a higher prioritized job becomes ready. Additionally, the scheduler aborts a job at the end of the reservation time or at the end of the period and the job (or part) is not accounted for the successfully executed optional parts.

3. QAS for Preemptible Resources

First, we formalize the task model concerning preemptible resources. After that we will describe QAS in three steps: task sets with uniform periods, harmonic periods, and arbitrary periods. Each step includes the discussion of priority assignment and the admission criterion.

3.1. Task Model

Each task T_i is a sequence of jobs J_{ij} to be processed periodically:

$$T_i = (J_{ij})_{j=1,2,...} \quad i = 1,...,n \quad (3)$$

where $n \in \mathbb{N}$ denotes the total number of tasks in the task set $\boldsymbol{T} = \{T_1,...,T_n\}$. Each job consists of one mandatory part M_{ij} and one optional part O_{ij}. M_{ij} is released at the beginning of its period, O_{ij} becomes ready when M_{ij} is completed. The period-end is the relative deadline of both parts. The execution time of the parts may vary (the mandatory parts do not exceed the WCET w_i), described by random variables. We assume that all random variables of all tasks are pairwise independent, and for each task T_i the random variables describing the mandatory parts are assumed to be identically distributed as well as for all optional parts. Finally, an application may specify a percentage q_i of optional parts that have to be completed successfully. In summary, the following definition describes a task.

Definition 1. A task T_i is a tuple

$$T_i = (X_i, Y_i, w_i, q_i, d_i)$$

where

- X_i nonnegative random variable; execution time of the mandatory part;
- Y_i nonnegative random variable; execution time of the optional part;
- w_i nonnegative real number less or equal to d_i; worst case execution time of the mandatory part;
- q_i real number $0 \leq q_i \leq 1$; quality parameter, probability that an optional part is completed;
- d_i positive real number; period length = relative deadline.

For simplicity, we identify the parts with their random variables, considering each mandatory part M_{ij} as a realization of X_i and each O_{ij} as a realization of Y_i. Obviously, $Y_i \equiv 0$ enables us to model tasks consisting of mandatory parts only (likewise for optional parts only).

The admission goal is to derive the "output parameters", namely the priorities $pr(X_i)$ and $pr(Y_i)$ of mandatory and optional parts and the reservation time r_i from the "input parameters" listed above to generate a *feasible schedule*, which means that all mandatory parts meet their deadlines and all optional parts meet their quality requirements.

3.2. QAS for Task Sets with Uniform Periods

In this section, we will describe the priority assignment and the admission criterion formula in the case of tasks with uniform periods, i.e.

$$d_i = d \quad \forall i = 1,...,n. \quad (4)$$

3.2.1. Priority Assignment. Since each mandatory part precedes its optional part and must meet its deadline even in worst-case situations, we give X_i an arbitrary but high priority. For the priority assignment of optional parts we introduced "Quality-Monotonic Scheduling" (QMS) in [9] analogous to the well known rate monotonic scheduling RMS: the higher the quality, the higher the priority. Additionally, the priorities of the optional parts have to be lower than the priorities of the mandatory parts.

This priority assignment is claimed to be optimal in respect of feasibility. That means: if a feasible schedule does not exist under QMS then such a schedule does not exist under any other fixed priority assignment (using reservations). However, this was not proved. We scrutinized this claim for both preemptible and nonpreemptible resources if all tasks have uniform periods. Although we constructed some examples confirming the assumption, we could neither proof nor rebut the proposition. An exact proof based on transformations of the admission criterion formula fails due to the structure of this formula (see below). Moreover, the problem to find optimal schedules seems to be NP-complete already for uniform periods (similar to the scheduling of C-jobs with Imprecise Computations [5]). Up to now, a successful investigation failed due to a further reason: the problem does not match any of the well-known NP-complete problems [6].

As the conclusion, we use QMS as a *heuristic* priority assignment here ($\succ$ means higher for priorities):

$$\left.\begin{array}{l} pr(X_i) \succ pr(Y_j) \\ pr(Y_i) \succ pr(Y_j) \ \text{ if } \ q_i > q_j \end{array}\right\} \ \forall i,j = 1,...,n. \qquad (5)$$

3.2.2. Reservation Times. Since the mandatory parts are higher prioritized than the optional parts, Condition (A1) of the admission criterion obviously holds if and only if

$$\sum_{i=1}^{n} w_i \le d$$

or equivalent

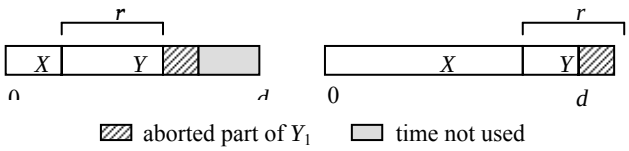
$$\sum_{i=1}^{n} \frac{w_i}{d} \le 1. \qquad (6)$$

To derive the reservation times r_i it is sufficient to consider the first period $[0, d]$ under the assumption that all tasks are ready at instant 0. Obviously, the reservation time must be smaller than the period length (i.e., $r_i \le d$ for all i). Due to the priorities of X_i the mandatory parts of all tasks T_i are scheduled before any optional part is scheduled. Hence, let X denote the sum of the execution times of all mandatory parts:

$$X = \sum_{i=1}^{n} X_i \ . \qquad (7)$$

Without loss of generality, we assume that $\boldsymbol{T}$ is ordered according to the priorities of the optional parts; so Y_1 (immediately scheduled and executed after the last mandatory part) has highest priority (but lower than the priority of any X_i) and so on. Thus, Y_1 is only successfully completed if it neither exceeds its reservation time r_1 within the period (Fig. 3(a)) nor it exceeds the period-end (Fig. 3(b)). So it follows

$$p_1(r) = \mathbf{P}(Y_1 \text{ is completed}) = \mathbf{P}(Y_1 \le r \ \wedge \ X + Y_1 \le d). \quad (8)$$



(**a**) (**b**)

Figure 3. Deriving the probability P(Y_1 is completed).

If $i \ge 2$, we have to take into account that each job Y_j, $j < i$, with a higher priority than Y_i has been started, independent of whether Y_j could be completed (Fig. 4(a)) or not (Fig. 4(b)), but Y_j was aborted when it reached its reservation time r_j. Thus,

$$p_2(r) = \mathbf{P}(Y_2 \le r \ \wedge \ X + \min(Y_1, r_1) + Y_2 \le d). \qquad (9)$$

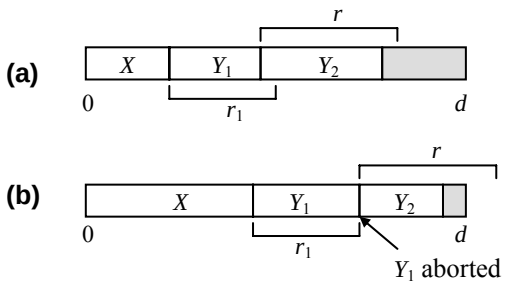


Figure 4. Deriving the probability P(Y_2 is completed).

In general, the probability $p_i(r)$ that an optional part O_i is completely executed within a given reservation time r is defined by:

$$p_i(r) = \mathbf{P}(Y_i \le r \ \wedge \ X + \sum_{j=1}^{i-1} \min(Y_j, r_j) \ + Y_i \ \le \ d),$$
$$i = 2,...,n. \qquad (10)$$

Now we can successively solve the system of equations (2) for all optional parts. To find a numerical solution to (10), we assume that all X_i, Y_i are discrete random variables; their values are natural numbers (as well as the period lengths). This assumption specifically holds whenever the random variables are given by an empirical distribution law resulting from measurements. Otherwise we substitute a continuous distribution function by a sufficiently precise decomposition of the set of values into disjoint classes. If we write the equations in a recursive form, which is more suitable for the numerical treatment, it results from (2) and (10) due to the laws of the probability algebra:

Theorem 1. For a given task set $\boldsymbol{T} = \{T_1,...,T_n\}$ with $d_i = d$ for all i the reservation times r_i can be calculated as

$$r_i = \min\left(r \in [0,d] \mid \sum_{k=0}^{r} \mathbf{P}(A_i \le d-k) \cdot \mathbf{P}(Y_i = k) \ge q_i \right),$$

$$i = 1,..., n \qquad (11)$$

where

$$A_1 := X = \sum_{i=1}^{n} X_i ,$$

$$A_i := A_{i-1} + \min(Y_{i-1}, r_{i-1}), \quad i = 2,..., n. \qquad (12)$$

□

The computation of $\mathbf{P}(A_i \le d - k)$ requires the repeated convolution of distribution functions because of (7) and (12). Since the probabilities $p_i(r)$ obviously increase monotonically with time r, we can use binary search to find the reservation times.

3.3. Harmonic Periods

Now we will remove the limitations of the original QAS admission model. First, we consider task sets with harmonic periods, i.e., any longer period must be an integer multiple of all shorter periods:

$$d_i < d_j \rightarrow d_i \mid d_j \quad \forall i,j = 1,...,n. \qquad (13)$$

3.3.1. Priority Assignment. For the reasons described in Sect. 3.2.1, we choose the following heuristic priority assignment denoted as *Extended QMS (EQMS)*. The task set $\boldsymbol{T}$ is decomposed into m disjoint subsets $\boldsymbol{T}_1,...,\boldsymbol{T}_m$; a subset consists of all tasks with the same period length d_i ($i = 1,...,m$). The subsets are ordered according to period length ($\boldsymbol{T}_1$ contains the shortest periods and so on, see Fig. 5). Now priorities are assigned according to QMS to both the mandatory and the optional parts of $\boldsymbol{T}_1$. After that we treat $\boldsymbol{T}_2$ just the same way, but the highest assigned priority must be lower than any priority of $\boldsymbol{T}_1$, and so on. A simple example below will show that this priority assignment is not optimal.

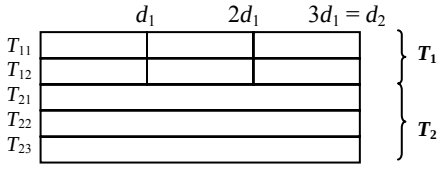


Figure 5. Task set structure.

3.3.2. Admission Criterion and Reservation Times. We use the following notations ($i = 1,...,m,\ j = 1,...,n_i$):

$\boldsymbol{T_i} = \{\boldsymbol{T}_{i1},..., \boldsymbol{T}_{i,n_i}\}$	i^{th} task subset
d_i	uniform period length of the tasks in $\boldsymbol{T_i}$
q_{ij}	quality parameter of task T_{ij}
X_{ij}, Y_{ij}	mandatory resp. optional part of task T_{ij}
$X_i = \sum_{j=1}^{n_i} X_{ij}$	execution time of all mandatory parts of subset $\boldsymbol{T}_i$
w_{ij}	WCET of X_{ij}
r_{ij}	reservation time of Y_{ij}
$n*X$	n-times sum of the random variable X ($n \in \mathbb{N}$).

To check Condition (A1) of the admission criterion, we consider the worst case (see Fig. 6): all mandatory parts consume their WCET, all optional parts completely use their reservation time. Then, the task set $\boldsymbol{T}$ can be admitted with respect to the mandatory parts if and only if the following condition holds:

$$\sum_{k=1}^{i-1} \frac{d_i}{d_k} \cdot \sum_{l=1}^{n_k} \frac{w_{kl} + r_{kl}}{d_k} + \sum_{k=1}^{j} \frac{w_{ik}}{d_i} \le 1$$

$$\forall i = 1,...,m \ \forall j = 1,...,n_i. \qquad (14)$$

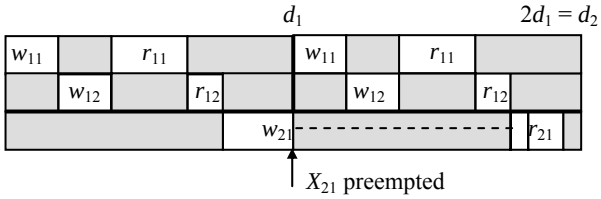


Figure 6. Admission of mandatory parts.

To derive the reservation time of an optional part, we have to take into consideration:

(a) mandatory parts of a task with longer periods are lower prioritized than optional parts of a task with shorter periods,

(b) optional parts are aborted by the end of the reservation time,

(c) optional parts are aborted by the end of the period,

(d) a period of the length d_i contains d_i/d_k periods of the shorter period of length d_j.

Thus, we have

Theorem 2. For a given task set $\boldsymbol{T}$ with harmonic periods, the reservation times r_{ij} under EQMS priority assignment can be calculated as follows:

$$r_{ij} = \min(r \in \mathbb{R} \mid \boldsymbol{P}(Y_{ij} \le r \wedge A_i + B_i + Y_{ij} \le d_i) \ge q_{ij})$$

$$\forall i = 1,...,m \ \forall j = 1,...,n_i \qquad (15)$$

where

$$A_i = \sum_{k=1}^{i-1} \frac{d_i}{d_k} * \min\left(d_k, X_k + \sum_{l=1}^{n_k} \min(Y_{kl}, r_{kl}) \right) \qquad (16)$$

execution time of all subsets $\boldsymbol{T_k}$ with $k < i$ ($A_1 := 0$),

$$B_i = X_i + \sum_{k=1}^{j-1} \min(Y_{ik}, r_{ik}) \qquad (17)$$

execution time of all parts of T_i higher prioritized than Y_{ij}.

□

The term d_i/d_k in A_i results from property (d) listed above, the min-operations from properties (c) and (b), respectively.

As mentioned earlier, EQMS is a heuristic priority assignment. The following example shows that EQMS is not optimal.

Let $\boldsymbol{T} = \{T_1, T_2\}$ where X_1, X_2, Y_1 identically uniformally distributed like Z :

Z	1	2
p	0.5	0.5

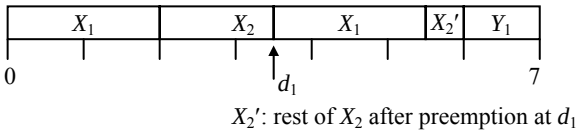
and

$$Y_2 \equiv 0 \quad d_1 = 3.5 \quad d_2 = 7 \quad q_1 = 0.4;$$

then the reservation time for Y_1 is $r_1 = 1$;
admission test for X_2:

$$\frac{w_1 + r_1}{d_1} + \frac{w_2}{d_2} \le 1 \qquad \text{fails since } \frac{8}{7} > 1 .$$

However, the task set is schedulable if the priority of X_2 is higher than the priority of Y_1, because then Y_1 can be scheduled at least in every second period (see Fig. 7).



X_2': rest of X_2 after preemption at d_1

Figure 7. Worst-case scenario for the example task set.

It should be mentioned that assigning all mandatory parts higher priorities than all optional parts (the latter according to EQMS) is also not optimal.

3.4. Arbitrary Periods

To derive a formal model for arbitrary periods and to explain the increased complexity of such a model, we consider an example first. The reason for this increase is the existence of "overlapping periods" (see Fig. 8): such a period ends in another period of the next lower prioritized task than it begins in.

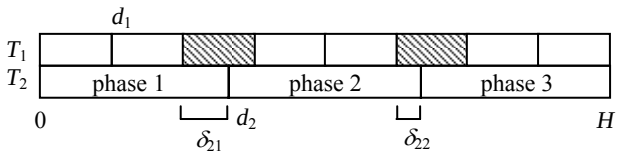


Figure 8. Overlapping periods ▨ and task phases for a task set with nonharmonic periods.

For clarity we assume $d_1 < d_2 < \ldots < d_n$ throughout this section. Hence, priorities *pr* are simply assigned according to $pr(X_1) \succ pr(Y_1) \succ pr(X_2) \succ pr(Y_2) \succ \ldots$.

Following the common terminology in real-time scheduling theory, the *hyperperiod* of two tasks T_1, T_2 with periods $d_1 < d_2$ is the interval $[0, H]$ where $H = \mathrm{lcm}(d_1, d_2)$. The hyperperiod contains H/d_2 *phases* of periods of task T_2. The time span available for task T_2 within its first phase Φ_1 is determined by the value of the random variable

$$Z_{21} = \left\lfloor \frac{d_2}{d_1} \right\rfloor * \left(\min\left(d_1, X_1 + \min\left(Y_1, r_1\right)\right)\right) + \\ + \min\left(\delta_{21}, X_1 + \min\left(Y_1, r_1\right)\right), \tag{18}$$

$$\delta_{21} = d_2 - d_1 \cdot \left\lfloor \frac{d_2}{d_1} \right\rfloor .$$

The min-operations reflect the actual scheduling behavior (see Fig.8):

$\min(Y_1, r_1)$: the optional part Y_1 is aborted at the end of its reservation time r_1;

$\min(d_1, \ldots)$: the execution of task T_1's first job is aborted at the end of its period d_1;

$\min(\delta_{12}, \ldots)$: the amount of time an overlapping job exceeds phase Φ_1 does not restrict the time span available for task T_2's first job.

Finally, $\left\lfloor \frac{d_2}{d_1} \right\rfloor$ is the number of complete (non-overlapping) jobs of task T_1 within phase Φ_1.

Thus, the corresponding reservation time r_{21} (task T_2 within its first phase Φ_1) is

$$r_{21} = \min(r \in \mathbb{R} \mid \mathbf{P}(Y_2 \le r \wedge X_2 + Y_2 + Z_{21} \le d_2) \ge q_2) . \tag{19}$$

To derive the next reservation time r_{22} (task T_2 within its second phase Φ_2) we have to take into account:

- all the jobs executed in non-overlapping periods during phase Φ_2; their number is
$$\left\lfloor \frac{2d_2}{d_1} \right\rfloor - \left\lceil \frac{d_2}{d_1} \right\rceil ;$$
- the potentially remaining portion of task T_1's overlapping job at the beginning of phase Φ_2 which is
$$\max(0, \min(d_1, X_1 + \min(Y_1, r_1)) - \delta_{21});$$
- the portion of task T_1's overlapping job at the end of phase Φ_2 during the time intervall of length
$$\delta_{22} = 2d_2 - d_1 \cdot \left\lfloor \frac{2d_2}{d_1} \right\rfloor .$$

In the last step (phase Φ_3 in the example), we either successfully get all reservation times or we can compute the maximal achievable quality during each phase of task T_2. Since it may occur that the admission of Y_2 fails within one (or more) of the phases but within another phase there is enough time to reach more than the requested quality, an equalization process should be done.

The final formula describing the general case uses the following notations:

- $H = \mathrm{lcm}(d_1,\ldots,d_n)$ hyperperiod length
- $h_i = \frac{H_i}{d_i}$ number of phases of task T_i
- $\Delta_{ij}^{(k)} = \left\lfloor \frac{kd_i}{d_j} \right\rfloor - \left\lceil \frac{(k-1)d_2}{d_1} \right\rceil$
- $\delta_{ij}^{(k)} = kd_i - d_j \cdot \left\lfloor \frac{kd_i}{d_j} \right\rfloor$
- $A_i = X_i + \min(Y_i, r_i)$
- $B_i = \min(d_i, A_i)$ actual resource usage of task T_i within a period
- $Z_{ij}^{(k)} = \Delta_{ij}^{(k)} * B_j + \min(\delta_{ij}^{(k)}, A_j)$
- $p_{ik}(r)$ probability that task T_i's job in phase Φ_k is completely executed within its period and time r
- r_{ik} reservation time of task T_i's job in phase Φ_k
- $x \dot{-} y = \max(0, x - y)$ nonnegative subtraction

where

$i = 1,\ldots,n$ current task index
$j = 1,\ldots,i-1$ lower prioritized task index
$k = 1,\ldots,h_i$ phase index.

Then we summarize:

Theorem 3. For a given task set $\boldsymbol{T} = \{T_1,...,T_n\}$ with $d_1 < d_2 < \ldots < d_n$ and $pr(X_1) \succ pr(Y_1) \succ pr(X_2) \succ \ldots pr(Y_n)$ the reservation times result from

$$p_{i1}(r) = \mathbf{P}\left(Y_i \le r \wedge X_i + Y_i + \sum_{j=1}^{i-1} Z_{ij}^{(1)} \le d_i \right),$$

$$p_{ik}(r) = \mathbf{P}\left(Y_i \le r \wedge X_i + Y_i + \sum_{j=1}^{i-1} \left(Z_{ij}^{(k)} + \left(B_j \dot{-} \delta_{ij}^{(k-1)} \right) \right) \le d_i \right)$$

$$i = 1,\ldots,n, \quad k = 2,\ldots,h_i. \qquad (20)$$

□

Due to the priority assignment, we can compute the reservation times for each task individually in descending priority order. Each step includes the admission test for the mandatory parts, for which we use time-demand analysis [13] (or another necessary and sufficient criterion) with a slight modification: the reservation time must be added to the WCET of all lower prioritized tasks.

As above (Sect. 3.2.2.), we apply a binary search algorithm to calculate the reservation times. It may occur that r_{ik} does not exist due to the restriction by the period length since a low prioritized task requesting a high quality is released late within its period. Then the algorithm yields the largest achievable quality $\tilde{q}_i$ and the corresponding reservation time $\tilde{r}_{ik}$ so that $p_{ik}(\tilde{r}_{ik}) = \tilde{q}_i < q_i$. Now we have three alternatives: we can

- abort the admission procedure because the given task set $\boldsymbol{T}$ is not schedulable;
- accept the lower quality and continue the algorithm;
- try to compensate the loss of quality in later (or earlier) phases of task T_i.

The latter makes sense because each task is assigned a tuple of reservation times corresponding to its phases. However, this tuple is not uniquely determined since the requested qualities can be modified (for instance, (70%, 70%) and (60%, 80%) results in the same overall quality). The consequence is a fnite but expensive iterative equalization process.

If some of the tasks have the same period length we use EQMS again. The only modification of the reservation time algorithm is to adapt the execution time of the mandatory parts reflecting their higher priority than that of optional parts inside a task subset $\boldsymbol{T}_i$: X_i is substituted by $\sum_{l=1}^{j-1} X_{il}$ to compute r_{ij}.

The equalization process mentioned above is one of the reasons why we did not implement the model described in this section. A more important reason are the enormously increased costs to compute the reservation times because the hyperperiod may become very large even for small task sets with close-by period lengths (like 503 and 510) and all phases must be considered. So in future work we will look for a different approach to overcome this difficulty.

4. QAS for Task Sets Using Nonpreemptible Resources

First, we summarize the QAS approach for nonpreemptible resources described in [9] in the case of uniform periods. Thereafter, we will extend it on harmonic and arbitrary periods.

4.1. Uniform Periods

To make the basic ideas of QAS with respect to disk requests applicable, the optional parts of each task T_i are divided into a fixed number m_i of *subjobs* (otherwise all parts of a task achieve a quality of either 100% or 0%). Y_i now describes the execution time of such a subjob. Priorities are assigned according to QMS; the task set has to be ordered according to these priorities. The reservation time r_i is determined for the optional parts as a whole of task T_i. During this time, a varying number of

subjobs can be started within each period. S_i (random variable) denotes this number. Due to the nonpreemptibility, each subjob that is started is successfully completed. q_i denotes the requested percentage of such subjobs (not optional parts!). Thus the admission criterion is

$$\exists r_1,\ldots,r_n \in \mathbb{R}\ \forall i = 1,\ldots,n:$$
$$r_i = \min(r \in \mathbb{R} \mid \mathbf{E}S_i \geq q_i m_i) \qquad (21)$$

where

$S_i = S_i(r,r_1,\ldots,r_{i-1})$ number of completed subjobs of task T_i within a period

$$\mathbf{E}S_i = \sum_{k=1}^{m_i} k \cdot \mathbf{P}(S_i = k)$$

expected value of the random variable S_i.

To derive the reservation time formula, random variables U_{ik} are defined describing the time to execute exactly k subjobs of an optional part of task T_i ($k = 1,\ldots,m_i$) respecting the fact that a started subjob cannot be aborted and so it may exceed its reservation time (see Fig. 9). Then the total execution time U_i is the sum of U_{ik} weighted by the corresponding probabilities:

$$U_i = \sum_{k=1}^{m_i} P(S_k = k) \cdot U_{ik}\,. \qquad (22)$$

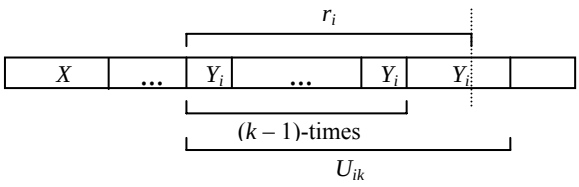


Figure 9. Reservation time r_i and execution time U_{ik} of an optional part of task T_i.

The approach presented in [9] ignores "pending" subjobs. Such jobs reduce the next period P of task T_i for two reasons: a subjob of T_i exceeds the end of its own period or a lower prioritized subjob overlaps P. We can handle this situation using an accurate but complex model or easily avoiding such subjobs. For the latter, we introduce a subjob WCET $w_{O,i}$ now. We diminish the period of T_i by $w_{O,i}$ and the longest WCET of all lower prioritized optional parts for the admission control:

$$d_i' = d_i - w_{O,i} - \max_{j>i}(w_{O,j})\,. \qquad (23)$$

This results in a slightly lower resource utilization but avoids the pending subjobs.

4.2. Harmonic and Arbitrary Periods

For harmonic periods, again we assign priorities according to EQMS. Within a task subset $\boldsymbol{T}_i$, the admission control is done as described for uniform periods, but considering the time which is consumed by the task subsets with shorter periods. This time is calculated based on U_i (Eqn. (22)) and the ratio of the period lengths. Similar to the pending subjobs at the end of a period described above, the execution of a task with a shorter period can be delayed due to the execution of a subjob of a task with a longer period. This is already solved by the inclusion of the subjob WCET $w_{O,i}$. We apply the same procedure for arbitrary periods.

5. Evaluation

We use both simulations and measurements to evaluate the accuracy of our admission models. The simulations generate a sequence of jobs with execution times based on the distributions used to calculate the reservation times. At the end of this section we will investigate complexity and on-line overhead of the QAS approach.

5.1. Preemptible Resources

To demonstrate the almost full compliance of the requested quality q and the achieved quality q_{ach}, we show two simulation experiments. The first experiment uses modified normal distributions (truncated at 0 and at the WCET). Table 1(a) shows the parameter and results of the task set $\boldsymbol{T} = \{T_{11}, T_{12}, T_2\}$. The second experiment uses empirical distributions. The values of X_{11}, X_{12}, and X_2 vary between 1 and 20, those of Y_{11}, Y_{12}, and Y_2 between 1 and 15. The period lengths are $d_1 = 65$, $d_2 = 195$. Table 1(b) shows the requested and the achieved qualities as well as the reservation times.

Table 1. Experimental results based on (a) normal distributions

Task	X: μ σ	w	Y: μ σ	q	d	r	q_{ach}
T_{11}	4 1	5	3 1	**0.70**	20	3.52	**0.7001**
T_{12}	3 2	6	2 1	**0.50**	20	2.00	**0.5016**
T_2	6 3	10	9 6	**0.91**	60	19.04	**0.9101**

(b) empirical distributions

Task	q	r	q_{ach}
T_{11}	**0.75**	13	**0.7500**
T_{12}	**0.57**	5	**0.5699**
T_2	**0.84**	9	**0.8399**

5.2. Nonpreemptible Resources

We chose disk drives to evaluate the admission model for nonpreemptible resources, because disk requests cannot be aborted once they are sent to the disk drive. A disk request can be mapped to a subjob of a task part, so the mandatory and optional parts of a task describe data streams read from or written to the disk. Assuming a fixed request size, the number m_i of subjobs of a part determines the bandwidth of the data stream. The execution time variance of a disk request is caused by the disk drive. Fig. 10 shows the distribution for the disk we used in our experiments.

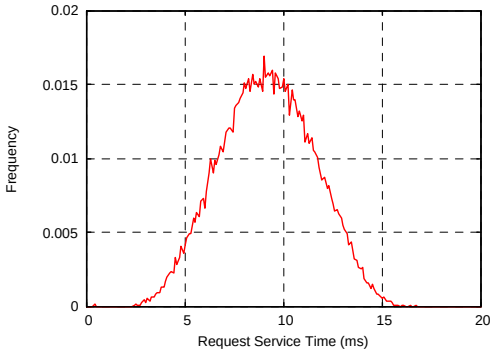


Figure 10. Request service time distribution (Seagate Cheetah 36ES disk drive; WCET = 40ms; 32 KByte read requests; random, uniformly distributed workload).

To evaluate the admission model, we implemented a SCSI resource scheduler in the Dresden Real-Time Operating System DROPS based on the scheme of cooperating resources managers [11]. The disk scheduler of our system periodically executes the disk requests of its clients and measures the execution time of each disk request. Requests were executed as long as the client did not exceed its time budget for the period.. The budget is set to the reservation time at the beginning of each new period.

Table 2 shows the task set we used in the evaluation. It is a "maximum" task set, meaning that it fully utilizes the disk drive according to the admission control. The last row shows the bandwidth achieved by a best-effort stream executed in parallel to the real-time streams. The bandwidths represent typical values for compressed video and audio streams. The results of the measurement as well as the simulation demonstrate that we achieve the requested qualities except for the last stream, which achieves a higher quality. This difference is caused by our approximation of the overlapping of requests at the period-end as discussed in Sect. 4.1. The results also show that we in fact fully utilize the disk drive, denoted by the low bandwidth (0.7%) that is left over for a best effort stream executed in parallel to the real-time streams, as shown in the last row of Table 2.

All the examples convince of the accuracy and feasibility of both models.

Table 2. Measurement and simulation results

Data stream	Period length (s)	No. subjobs	**Quality requ.**	Reservation time (ms)	**Quality measured**	Quality simulated	Bandwith meas. (KByte/s)
1	1	24	**1.00**	-	**1.000**	1.000	768.0
2	1	24	**0.95**	206.40	**0.953**	0.951	731.9
3	1	20	**0.95**	171.60	**0.954**	0.951	610.8
4	1	24	**0.85**	184.40	**0.863**	0.857	662.7
5	1	20	**0.85**	151.80	**0.856**	0.851	547.7
6	8	12	**0.95**	101.60	**0.946**	0.950	45.4
7	8	8	**0.95**	66.80	**0.952**	0.952	30.5
8	8	12	**0.90**	95.20	**0.903**	0.902	43.4
9	8	8	**0.87**	73.20	**0.957**	0.938	30.6
-	-	-	-	-	-	-	23.6

5.3. Computational Complexity and Admission Overhead

The computational complexity of the admission models for preemptible resources is dominated by the number of convolutions of distributions required to compute A_i, B_i, or $Z_{ij}^{(k)}$ in Eqn. (12), (16/17), or (20), respectively. Those terms must be calculated for each task of a task set. With discrete random variables X, the complexity of one convolution is v^2, where v is the number of values of X. The reservation time computation is solved using nested intervals which results in a complexity of $nv \cdot \log v$. Hence, the overall complexity for uniform, harmonic, and arbitrary periods is $O(n \cdot v^2)$, $O(n^2 \cdot v^2)$, resp. $O((n-1)! \cdot v^2)$ (the latter due to the consideration of the hyperperiod).

The complexity of the admission control for nonpreemptible resources is influenced by the costs to compute U_i in Eqn. (22). Their complexity is $O(s \cdot v^3)$ where s is the sum of all optional subjobs of the task set $\boldsymbol{T}$.

Table 3 shows the actual costs of the admission control for the first example task set used in Sect. 5.1 for various class sizes of the used distributions.

Table 3. Admission costs for task set used in example 1, see Table 1(a) (Pentium M 1.6 GHz)

Class size	0.10	0.05	0.01
Admission time (ms)	2.1	8.2	198.0

The admission for the task set used in Sect. 5.2 takes about 3s for a class width of 0.5ms. Finally, the run-time overhead caused by the scheduler (manipulations of the ready queue) is negligible independent of the type of resources and the type of periods.

6. Related Work

The existing literature offers a large amount of work on

- supporting a predictable system behavior in transient or permanent overload situations,
- providing statistical guarantees for firm real-time applications (which tolerate occasional misses of hard deadlines),
- considering of varying execution times to improve resource utilization,
- enforcing of time restrictions by a reservation based scheduler.

To the best of our knowledge, QAS is the only approach which achieves all these goals together. Moreover, QAS is not restricted for specific resources or applications in contrast to other methods.

The approach closest to QAS is Statistical Rate Monotonic Scheduling (SRMS) [3]. The idea is to assign an "allowance" (similar to reservation time) to a task during its "superperiod" (i.e., the period of the next lower prioritized task according to RMS). A local admission – executed at the release time of any job – assures each task a percentage of successful jobs. However, SRMS has some disadvantages and limitations: jobs cannot be divided in parts or subjobs, it cannot handle nonpreemptible resources, and above all the actual execution time must be known at the release time of each job.

The idea to divide jobs into mandatory and optional parts is taken from the Imprecise Computation Model (ICM) [5]. It attempts to minimize the total error in the results of jobs which need not be completed (*N*-jobs) or must be completed in one period among several consecutive periods (*C*-jobs). A class of preemptive, priority-driven scheduling algorithms is proposed but based on WCET, and not applicable for nonpreemptible resources. The same holds for other models of firm real-time systems. One of them is the (*m*,*k*)-firm tasks model [10] in which a task must meet at least *m* deadlines within a "window" of *k* consecutive invocations. In [2] the ICM approach is employed for managing of quality of service of real-time databases even for transient overload.

In a similar way TIA [17] et al. proposed a "transform task method" to provide probabilistic schedulability guarantees to semi-periodic real-time tasks where the ratio of maximum computation time to period is larger than 1. The authors transform each task into a periodic task followed by a sporadic task, comparably mandatory and optional parts. They compute the probability that each task will meet its deadline, and develop a probabilistic time demand analysis which substitutes the sums of fixed execution times with convolutions of probability density functions. The method demands that the exact computation time of each request (job) of a task becomes known when the task is released.

DIAZ et al. [7] describe a stochastic analysis method for a wide class of periodic real-time systems. The proposed method computes the response time distribution of each task based on a Markovian modeling, thus making it possible to determine the deadline miss probability of individual tasks. The computation of the complete probability function of the response time is similar to our approach (the "shrinking" operation corresponds to the min-operation). Several other papers propose analysis methods for real-time tasks with variable execution times and offer algorithms to compute deadline miss probabilities [1, 4, 8, 14, 15]. However, the presented approaches do not allow the governing of systems to achieve a requested system behavior.

In [12] the problem of temporal consistency maintenance of real-time data objects is studied where a certain degree of temporal inconsistency is tolerable. A transaction *T* periodically updates such an object. The job admission of *T* guaranteeing a requested percentage of updates (quality) is based on the quantile of the computation time distribution of *T*. However, that is a *local* admission without CPU reservation. The admission demands to know the job computation time a priori (similar to SRMS). Finally, the goal is to maximize the overall quality but not to produce feasible schedules.

Several approaches use statistical methods to improve the utilization of disk drives in multimedia servers. All of these methods aim to calculate the probability of deadline misses for a given workload, based on either a probabilistic model of the disk drive [16, 19] or the measured execution time distribution of disk requests [18]. In contrast to these approaches, QAS provides alogorithms to calculate the amount of resources that is required to achieve a requested quality. This allows the use of more flexible scheduling algorithms, for example the use of slack-stealing scheduling to include tasks not accounted by the admission control, such as sporadic and best-effort tasks.

7. Summary

This paper overcomes the limitations of the "Quality-Assuring Scheduling" approach presented in [9]. We give in detail the admission criteria and the formulae to compute the reservation times for preemptible and non-preemptible resources in the case of harmonic and arbitrary periods. Both simulation experiments and system measurements affirm the efficiency of the approach. Future work should reduce the high admission costs for task sets with nonharmonic periods.

References

[1] L. Abbeni and G. Buttazzo, "Stochastic Analysis of a Reservation Based System". In *Proc. of the 9th International Workshop on Parallel and Distributed Real-Time Systems,* April 2001.

[2] M. Amirijoo and J. Hansson, "Robust Quality Management for Differentiated Imprecise Data Services". In *Proc. of the 25th Real-Time Systems Symposium (RTSS'04)*, pp. 265-275,, December 2004.

[3] A. Atlas and A. Bestavros, "Statistical Rate Monotonic Scheduling". In *Proc. of the 9th International Workshop on Parallel and Distributed Real-Time Systems,* April 2001.

[4] G. Bernat, A. Colin, and S. Petters, "WCET Analysis of Probabilistic Hard Real-Time Systems". In *Proc. of the 23th IEEE Real-Time Systems Symposium (RTSS'02)*, Dec. 2002.

[5] J.-Y. Chung, J. W. S. Liu, and K.-J. Lin, "Scheduling Periodic Jobs That Allow Imprecise Results". In *IEEE Trans. on Computers*, vol. 39, no. 9, Sept. 1990.

[6] P. Crescenzi and V Kann, "A compendium of NP optimization problems". http://www.nada.kth.se/~viggo/problemlist/compendium.html

[7] J. L. Diaz, D. F. Garcia, K. Kim, C.-G.Lee, L. L. Bello, J. M. López, S. L. Min, and O. Mirabella, "Stochastic Analysis of Periodic Real-Time Systems". In *Proc. of the 23th IEEE Real-Time Systems Symposium (RTSS'02)*, Dec. 2002.

[8] M. K. Gardner and J. W. Liu, "Analyzing Stochastic Fixed-Priority Real-Time Systems". In *Proc. of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 44–58, March 1999.

[9] Cl.-J. Hamann, J. Löser, L. Reuther, S.Schönberg, J. Wolter, and H. Härtig, "Quality-Assuring Scheduling–Using Stochastic Behavior to Improve Resource Utilization". In *Proc. of the 22th Real-Time Systems Symposium (RTSS'01)*, pp. 119-128, December 2001.

[10] M. Hamdaoui and P. Ramanathan, "A Dynamic Priority Assignment Technique for Streams with (m,k)-Firm Deadleins". In *IEEE Trans. on Computers*, vol. 44, no. 12, pp. 1443-1451, December 1995.

[11] H. Härtig, L. Reuther, J. Wolter, M. Borriss, and T. Paul, "Cooperating resource managers". In *Fifth IEEE Real-Time Technology and Applications Symposium (RTAS)*, Vancouver, Canada, June 1999.

[12] K.-Y. Lam, M. Xiong, B. Liang, and Y. Guo "Statistical Quality of Service Guarantee for Temporal Consistency of Real-Time Data Objects". In *Proc. of the 25th Real-Time Systems Symposium (RTSS'04)*, pp. 276-285, December 2004.

[13] J. P. Lehoczky, L. Sha, and Y. Ding, "The rate-monotonic scheduling algorithm: Exact characterization and average case behavior". In *Proc. of Real-Time Systems Symposium*, pp. 166-171, December 1989.

[14] S. Manolache, P. Eles, and Z. Peng, "Schedulability Analysis of Applications with Stochastic Task Execution Times". *ACM Journal, Vol. 3, No. 4*, Nov. 2004.

[15] A. K. Mok and D. Chen, "A Multiframe Model for Real-Time Tasks". *IEEE Transactions on Software Engineering* vol.23, no. 10), pp. 635–645, Oct. 1997.

[16] G. Nerjes, P. Muth, and Gerhard Weikum, "Stochastic Service Guarantees for Continuous Data on Multi-Zone Disks". In *Proc. of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '97)*, pp. 150-160, May 1997.

[17] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.-S. Liu, "Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times". In *Proc. of the Real-TimeTechnology and Applications Symposium,* pages 164–173, May 1995.

[18] H. M. Vin, P. Goyal, A. Goyal, and A. Goyal, "A Statistical Admission Control Algorithm for Multimedia Servers". In *Proc. of the 2nd ACM International Conference on Multimedia (ACM Multimedia '94)*, pp. 33-40, October 1994.

[19] R. Zimmermann and K. Fu, "Comprehensive statistical admission control for streaming media servers". In *Proc. of the 11th ACM International Multimedia Conference (ACM Multimedia 2003)*, pp. 75-85, Nov. 2003.