SICS Software-Intensive Cyber-Physical Systems (2021) 35:201–236
https://doi.org/10.1007/s00450-021-00429-1

REGULAR PAPER

Beyond the limitations of real-time scheduling theory: a unified scheduling theory for the analysis of real-time systems

Frank Slomka[1] · **Mohammadreza Sadeghi**[2]

Received: 13 April 2021 / Accepted: 18 October 2021 / Published online: 29 November 2021

Abstract
We investigate the mathematical properties of event bound functions as they are used in the worst-case response time analysis and utilization tests. We figure out the differences and similarities between the two approaches. Based on this analysis, we derive a more general form do describe events and event bounds. This new unified approach gives clear new insights in the investigation of real-time systems, simplifies the models and will support algebraic proofs in future work. In the end, we present a unified analysis which allows the algebraic definition of any scheduler. Introducing such functions to the real-time scheduling theory will lead two a more systematic way to integrate new concepts and applications to the theory. Last but not least, we show how the response time analysis in dynamic scheduling can be improved.

Keywords Scheduling theory · Scheduling test · Response time analysis · Static scheduling · Dynamic scheduling · Unification of scheduling theory · Dirac delta · Heaviside function

1 Introduction

If we have a careful review of existing work in real-time scheduling theory, mainly two different approaches to satisfy the real-time capability of an embedded system exist: the bound test or, in more general, the utilization based approach[1] and the response time analysis. The bound tests tests compute the utilization of a hardware resource as the response time analysis focus on the behaviour of tasks. In system analysis, both approaches are helpful. However, looking at related work, the two approaches are different in a tiny detail: while the utilization based computations built on the floor operator, the response time analysis uses the ceiling operator. Nevertheless, looking closer to previous work leads to problems in formulating a utilization-based test for static scheduling and a response time analysis for dynamic scheduling. However, in the practical use of the scheduling theory, the analysis of static scheduling prefers the response time analysis while the analysis in dynamic scheduling prefers the utilization based test. The observation is that the mathematical expressiveness of both functions is limited: the floor and the ceiling operator does not support algebraic properties such as distributivity and commutativity. Besides, these operators are not analytical in the sense that calculus is not well supported. The work of [12] and [43] show the limitations on floor and ceil operators in the context of the real-time scheduling theory. Both papers postulate new analysis techniques if an event count with more mathematical expressiveness is known. From a practical perspective, real-time analysis work always covers one concrete problem, and the algorithms published are solving just this particular problem. Combining different ideas is difficult because the task models often change. Sometimes different algorithms are used to address different problems in the application of the theory. Ref. [5] discusses different real-time analysis methods to compute task response times to cover multiple issues in the automotive industry and find that different approaches are necessary to cover all aspects needed.

This paper presents an approach to address both problems directly. If we look at another domain in science and engineering, the problem of discrete and continuous behaviour was already addressed. In digital signal processing and digital control theory both worlds, the discrete and the continuous

✉ Frank Slomka
frank.slomka@uni-ulm.de

[1] Embedded Systems/Real-Time Systems, Ulm University, Ulm, Germany

[2] Inchron AG, Erlangen, Germany

[1] Some people differ between bound tests and demand bound tests. In this paper, we follow an application-based structure: the interest of developers in response times and the problem of finding the utilization of a processor as a step in design space exploration.

nature of systems are combined. The idea of this paper is to adapt mathematical models used in physics, signal- and control theory to the problem of real-time scheduling analysis. As a result, we present

- A new universal mathematical framework, which allows to replace geometric only proofs given by diagrams and known from previous work with new algebraic and analytical methods in an intuitive way.
- A new generic approach to formulate interfering tasks in different scheduling policies
- and therefore a unified formulation of the bound tests- and the response time analysis in static and dynamic scheduling based on just one equation.
- Additionally we adopt assumptions of the analysis of arbitrary deadlines to the analysis of response times in dynamic scheduled systems and found a deterministic and tighter analysis as in previous work.

2 Related work

Real-time systems are computer systems whose software must complete calculations within fixed deadlines. For this purpose, the algorithms of an application are split into individual tasks, and these tasks are independently executable. A system requires an operating system that generates predictable execution sequences to ensure a time-related response. If an operating system delivers predictable schedules a mathematical model can be derived and deadline compliance can be calculated. During the Apollo missions to the moon, the first today-like real-time computer was used for guidance and navigation (Apollo Guidance Computer, AGC, [33], p.221 ff). During this time, software engineers expect a task utilization of 80% will guarantee correct real-time behaviour of the AGC. However, on the 20th of July 1969, during the first human-crewed landing, the computer of the lunar module Eagle gave a program alarm at decent to the moon's surface. The computer had to be reset three times during the whole landing, and the mission was short before abort. A later analysis at NASA figured out, that a wrong real-time behavior and the missing of the deadline of a flight critical task led to the problem. Later on, a mathematical analysis of [32] showed that the assumption a utilization of 80% on static real-time scheduling (rate monotonic scheduling, RMS) resulted in missing deadlines. [32] showed that the utilization limit of a static real-time task set is dependent on the number of tasks, and in the limit on a large number of tasks is only 69%. However, while this limit is only sufficient and not necessary, it was necessary to develop further real-time tests. While [32] considered the utilization of a task set in static and dynamic scheduling, other researchers followed an different approach, computing the response times of all tasks of a task set, as given by [27]. Since both, [32] as well as [27] assumed implicit deadlines defined by the period of events, [30] showed that deadline monotonic scheduling DMS) was the optimal priority assignment when the deadline is smaller than the period and [29] introduced a schedulability test on given checkpoints to DMS.

This first work in real-time scheduling theory were limited to uni-processor systems. An extention to distributed systems gives [49] by introducing a jitter based periodic event model. Later on, the response time analysis was generalized by [40] to integrate more complex event models. The response time analysis as given by [29] is limited to systems with static priorities. The extension for dynamically scheduled (earliest deadline first, EDF) real-time systems [38,39] needs to distinguish between different dynamic cases during analysis. This makes the approach complex. The real-time analysis distinguishes between load analysis (processor load) [32] and response time analysis [29]. Therefore, both directions are discussed independently in literature. The utilization based approach was extended and improved by [9], who introduced deadlines shorter than periods to the bound tests analysis of dynamic scheduling. However, this work supports only the periodic and sporadic event model wich does not allow the formulation of bursty events. A more general approach to model different and complex worst-case event patterns was first introduced by [23]. This event stream model could be very easy combined with Baruah's approach as shown by [3]. Because the analysis algorithm has a bad run-time complexity some approximations are introduced by [3] and [4] for dynamic scheduling and for static scheduling [21]. While the work of [23] does not model event bursts in an appropriate way, [1] introduce hierarchical event streams. Additionally, [24] use Baruahs utilization based scheduling test to design a novel response time analysis for dynamic scheduling. Other extensions are the multiframe- [34], the generalized multiframe [8] and the reccurring real-time task model [6]. These techniques allow the modeling of periodic task sequences with jobs with different execution times and extend real-time scheduling theory to the domain of stream processing systems [7,35] and with the most powerful model of [42].

In addition to these works, which are assigned to the classical theory of real-time systems (scheduling theory), the real-time behaviour of task systems can also be verified with the real-time calculus (RTC). The real-time calculus is based on the network calculus [13,18,19] which describes a mathematical framework for analyzing the flow of data in networks. [36,46] introducing the real-time calculus and apply their work [16,44,45] to the analysis of network processors. It was shown that the classical methods can be replaced by the real time calculus. In contrast to prior work, the real-time calculus allows the calculation of systems with many different scheduling strategies as static- (DMS) and dynamic scheduling (EDF), time-division multiplex access (TDMA)

and others. While the approach is modular it also allows hierarchical scheduling. Finally, by [28,40] response time analysis as given by the classical theory were combined with real-time calculus to build an analysis that highlights the strengths of each technique. The disadvantage of this work is that the modelling is not generic and must be redefined for each system to be modelled.

However, the existing work is split in utilization based techniques, response time analysis and the real-time calculus. Each approach has its advantages and disadvantages. Sometimes authors like to combine the different work but often they are missing event bound functions with different properties as given by the established theory. The need for new approaches is given in [12,24,43]. Other authors prefer an analysis technique independent from the application structure [28,40].

A way to introduce analytical proofs in real-time scheduling theory is presented in [14]. This work is limited to the busy window approach while the goal of the presented work is to combine utilization based test with the response time analysis. Because our new approach uses advanced techniques given by theoretical physics and signal theory, it is more compact and expressive than previous work in the real-time domain. Because of its expressiveness, it allows the formulation of a closed algebraic method. This method is open to different problems in real-time analysis. Such an approach leads to an easy formulation of utilization based and response-time based analysis in static as well as in dynamic scheduling. The idea allows a straightforward combination of both scheduling techniques without the overhead to formulate different equations and algorithms. It combines different event models and gives a new approach to the response time analysis of dynamic task systems. For the first time in literature, we present an approach that allows an explicit function to describe different schedulers.

3 Model of computation

Different computational models to analyze real-time systems exist. In this work, we consider the bounded execution time model. We are assuming that the execution flow in real-time systems separates into different tasks. A task is a kind of programming function assigned to an external or internal interrupt - an event - of the system. The tasks are periodically time- or event-controlled. Each event requests a task, and the concrete instance which occurs is called a job. Each job must be executed in a limited time interval: the deadline. In the bounded execution time model, tasks are preempted by higher priority tasks. The priority of the execution of a job can be assigned statically or dynamically. Bounding jobs of a task to a deadline allows any scheduling permutation without any sophisticated scheduling algorithm. In static scheduling, like rate monotonic/deadline monotonic (RMS/DMS) scheduling, the priorities are assigned statically to each task depending on the request rate of the triggering events. In dynamic scheduling, like the earliest deadline first (EDF) policy, the priority of each job depends on the next approaching deadline. Therefore, the scheduling priority is not strictly assigned to tasks. In the classical scheduling theory by [32], the bound tests of a task set in the sense if deadlines met, is proved by computing the utilization of resources like processors or the maximal response time of any worst-case job.

3.1 Events

A timing relationship between events is needed to compute a task set's utilization or the response time of the worst-case job or all other jobs as well. The established model defines a sequence of periodic events and the distance in time between events is denoted by a single value: the period $p \in \mathbb{R}_0^+$. Because each task has different periods, a function $p_\tau := p(\tau)$ may always return the period of the considered task. This event model has been extended to the sporadic event model where the period interprets as minimal inter-task arrival time. The periodic event model with jitter allows considering distributed systems in holistic real-time analysis [47,49]. This model was extended to include task offsets [51] and arbitrary deadlines to the response time analysis [48]. However, a more general model on events was first introduced by [23], has limitations to express bursty event patterns. Hierarchical event streams give a shorthand formulation to solve this problem. In this paper, we consider the periodic or sporadic event model and the event stream model in parallel. The periodic model in this work is used to give the reader a simple link to previous work, while the event stream model is more general and includes all derivates like the sporadic, the bursty, or the periodic model with jitter.

Definition 1 (*Event stream*) An event stream is an array of event tuples or an event list. Each event tuple ϵ describes a periodic sequence of events ϵ':

$$\mathcal{E} = \left\{ \begin{pmatrix} p_\epsilon \\ \phi_\epsilon \end{pmatrix} \right\} \tag{1}$$

The event stream must be valid, which means the order of the time intervals ϕ_ϵ must be subadditive or superadditive. If the event list does not fulfil the requirement of subadditivity, we call it an event sequence.

An event tuple consists of the period p of an event sequence and a minimal distance ϕ to another event. The position of the event tuple in the stream array has a meaning: The first tuple initializes the stream. It always has $\phi = 0$. The second tuple describes the minimal distance between two events, the third between three events and so on. This

means the interval given in each list element contains all previous events. Therefore each tuple represents the minimal distance of the related number of events and its periodical repetition.[2] In this work, each event sequence is indexed by ϵ. Therefore, p_ϵ denotes the period of event sequence ϵ and ϕ_ϵ the minimal distance ϕ between n-events. The number n of the events is given by the position of the event tuple in the event stream. Note, that in this model sporadic events can be described easily: an event which occurs only once has an infinite period.

Example 1 (Event model: periodic) Assume an event which occurs periodically every p time:

$$\mathcal{E}_{periodic} = \left\{ \begin{pmatrix} p \\ 0 \end{pmatrix} \right\} \tag{2}$$

The minimal distance of the first initial event is $\phi = 0$. The event recurs with the period p.

In the end, to formulate schedulability tests, a bound function to event streams is required:

Definition 2 (*Right-continuous event bound*) Assume any event stream as given by Definition 1, the event bound function or event bound $\overset{\bullet\circ}{\mathbb{E}}: \tau \times \mathbb{R} \rightarrow \mathbb{N}$ of any task is given by

$$\overset{\bullet\circ}{\mathbb{E}}_\tau (t) = \sum_{\epsilon \in \mathcal{E}_\tau} \left\lfloor \frac{t - \phi_{\tau,\epsilon}}{p_{\tau,\epsilon}} + 1 \right\rfloor \tag{3}$$

Final, the event bound of a task set is the sum of all independent task bounds.[3]

3.2 Tasks

The inter-arrival pattern of events only describes the occurrence of events. At each event, an independent part of a program is executed by the operating system. Such an execution unit is called a task τ. A real-time application separates into several tasks. Therefore each task is an element of a task set: $\Gamma := \{\tau_1, \tau_2, \ldots, \tau_n\}$. All tasks must schedule on the given processor in a way that all deadlines met. A scheduler is optimal if no algorithm exists, which produces a better valid schedule. In [32] was proven that RMS is optimal for static, and EDF is optimal for dynamic scheduling. Therefore an execution time must be added to the model. Because the execution of a task's job varies and we are only interested in worst-case bounds [32]. In the real-time analysis, a task is defined by an inter-arrival pattern of events and the two execution times. In the bounded execution model, the relative deadline specifies the time a task has to finish after being requested. If all tasks are independent, it is not necessary to consider the best case execution time. This parameter is only needed if tasks with data dependencies are running on different processors [22].

Definition 3 (*Execution time)* The execution time of a task is the time the execution of the task needs if a processor exclusively executes the task with no interruption by other tasks. The execution time may depend on data attributes given to the task. Therefore we distinguish between the worst-case or maximal (c^+, WCET) and best-case or minimal execution time (c^-, BCET).

As we consider the bounded execution model, a deadline must be assigned to each task. The deadline is a time interval in which the execution of a task must finish. It is distinguished between a relative deadline (d) and an absolute deadline (D).

Definition 4 (*Relative Deadline*) The relative deadline d_τ of a task bounds the execution of any job related to the request time t^r of this job.

Definition 5 (*Absolute Deadline*) The absolute deadline $D^n_{\tau,\epsilon}$ of the n'th job is related to $t = 0$. Therefore the n'th absolute deadline of the job is

$$D^n_{\tau,\epsilon} = \phi_{\tau,\epsilon} + np_{\tau,\epsilon} + d_{\tau,\epsilon} \tag{4}$$

During the execution of the task set, the operating system has to schedule jobs of the task set. The operating system determines the execution order of the jobs based on the relative or absolute deadline assigned to each job. In some cases, fixed priority numbers given by the programmer replacing deadline-based scheduling.

Definition 6 (*Static priority*) Let $\pi \in \mathbb{N}$ and assume two independent tasks τ and τ', a task τ' has a higher assigned priority than task τ, if $\pi_{\tau'} > \pi_\tau$ and assume a task with higher priority preempts tasks with lower priority. The set of all higher priority tasks of task τ is

$$\overline{\Gamma}_\tau := \{\tau' \in \Gamma \mid \pi_{\tau'} > \pi_\tau\} \tag{5}$$

Therefore, a task can be specified formally:

Definition 7 (*Task*) A task $\tau \in \Gamma$ is a quadruple including the inter-arrival pattern of events $\mathcal{E}$, the worst-case and best-case execution time of a task and a relative deadline d by which the task execution bounds:

$$\tau := \{\mathcal{E}, c^+, c^-, d, \pi\} \tag{6}$$

[2] Note, that the bounds of valid event streams can be interpreted as curves of the real-time calculus. Such a bound is defined in Eq. 40. Because of the subadditive or superadditive nature of valid event streams the critical instant theorem holds for any valid event stream.

[3] The function $p_{\tau,\epsilon} = p(\tau, \epsilon)$ defines a function which returns the period of an event sequence described by the event tuple ϵ of the task τ.

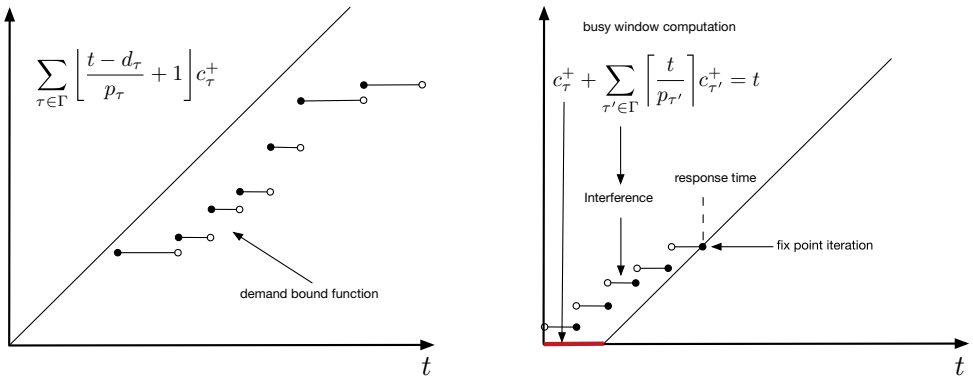


Fig. 1 Demand-bound- versus busy-window-test: the different properties of bounds

Note, that the relative deadline can be replaced or amended by a static priority π. Access to the data structure of a task can be granted by task dependent functions: $p_\tau = p(\tau)$, $c_\tau^+ = c^+(\tau)$,[4] etc.

In some work, to each job of a task different execution times assigned. In such a case, the execution times of a task are specified by a vector. Job-related execution times are introduced by the multi-frame task model [34]. If job-related deadlines added, this is called the generalized multi-frame model [8]. Therefore jobs must introduced in the task model:

Definition 8 (*Job*) A job is the instance of a task $\tau_\epsilon \in \tau$ triggered by any event of the event stream related to a task.

4 Motivation to a unified theory

Figure 1 summarizes the two approaches for the analysis of real-time systems. The demand bound test has to check if each value given by the demand bound is smaller than a given processor resource. Consider the left side of Fig. 1, to check for an intersection of the demand bound to the bisecting line left of the demand. We must consider each left bounded step of the demand bound function. Opposite to that approach, the response-time analysis checks for the intersection of the request bound with the processor resource. As seen on the right side of Fig. 1 only the left points must be checked for an intersection. Assuming a positive worst-case execution time of the considered task, this is the most right point of the request bound of all interfering tasks. Therefore, it is clear that both bounds need different mathematical properties.

4.1 Event bound approaches to real-time systems analysis

The demand bound function (dbf, $\overset{\bullet\circ}{\mathbb{D}}_\Gamma(t)$) is a composition of the deadline-shifted request bound function (rbf, $\overset{\bullet\circ}{\mathbb{R}}_\Gamma(t)$). On a given event bound function, the request bound is just the event bound multiplied by the worst-case execution time, while to construct the demand bound each deadline shifts this function to the right [9] as shown on the left side of Fig. 1:

$$\begin{aligned}\overset{\bullet\circ}{\mathbb{D}}_\Gamma(t) &= \sum_{\tau\in\Gamma}\overset{\bullet\circ}{\mathbb{R}}_\tau(t-d_\tau) = \sum_{\tau\in\Gamma}\overset{\bullet\circ}{\mathbb{E}}_\tau(t-d_\tau)c_\tau^+ \\ &= \sum_{\tau\in\Gamma}\left\lfloor\frac{t-d_\tau}{p_\tau}+1\right\rfloor c_\tau^+ \end{aligned} \tag{7}$$

It is necessary to count the number of interfering events of a task's job to compute the utilization of a resource or the worst-case response time. In other words, this means that the analysis adds the execution time of a job to the response time of the considered task if both interfere in the same time interval. Related work formulates variations of such a request bound to model the interfering of higher priority tasks.

Opposite to the event bound given in the bound tests test, in response time analysis, the event bound is given by a left continuous function [29]:

$$\begin{aligned} r_{\tau,n}^+ &:= c_\tau^+ + \sum_{\tau'\in\overline{\Gamma}_\tau}\overset{\circ\bullet}{\mathbb{E}}_{\tau'}(r_{\tau,n-1}^+)c_{\tau'}^+ \\ &= c_\tau^+ + \sum_{\tau'\in\overline{\Gamma}_\tau}\overset{\circ\bullet}{\mathbb{R}}_{\tau'}(r_{\tau,n-1}^+) \\ &= c_\tau^+ + \sum_{\tau'\in\Gamma}\left\lceil\frac{r_{\tau,n-1}^+}{p_{\tau'}}\right\rceil c_{\tau'}^+ \end{aligned} \tag{8}$$

What is the reason for the difference? As mentioned, the demand bound test has to check the left points of the demand bound while the busy window approach looks for an intersection on the right side of the request bound with the intersecting line. The initial value of the response time analysis gives the worst-case execution time of the considered task. In contrast to the demand bound test, the time point $t = 0$ does not matter because the minimal response time is if a task does not execute and always equal the best- or worst-case execution time. As this time interval is the starting point

[4] More general: $f_{k,l} = f(k,l)$.

of the fixed-point iteration, 0 never occurs in the equation. However, to find the intersection with the resource function, the bound must be left-continuous. Therefore, at the end of the busy interval, an event should not count. If a task finishes its execution and at the exact moment a new task requests, this request is superfluous.

It is obvious that $\overset{\bullet\circ}{\mathbb{E}}_\Gamma(t)$ is not equivalent to $\overset{\circ\bullet}{\mathbb{E}}_\Gamma(t)$, while $\left\lfloor \frac{0}{p_\tau} + 1 \right\rfloor = 1 \neq 0 = \left\lceil \frac{0}{p_\tau} \right\rceil$. However, in this work we want to consider the different properties of both functions to extend the theory and therefore we discuss them in more detail:

Lemma 1 (Identity) *The event bound functions* $\overset{\circ\bullet}{\mathbb{E}}_\Gamma(t)$ *and* $\overset{\bullet\circ}{\mathbb{E}}_\Gamma(t)$ *are not identical:* $\overset{\circ\bullet}{\mathbb{E}}_\Gamma(t) \not\equiv \overset{\bullet\circ}{\mathbb{E}}_\Gamma(t)$.

Proof To keep the proof simple, we only prove the lemma in the periodic case. Consider $t_n, t' \in \mathbb{R}^+$ and $\forall \tau \in \Gamma : 0 \leq t \leq p_\tau$. We investicate the three periodical repeating intervals $\Delta_{n-1,n} = (\{n-1\}p_\tau, np_\tau]$, $\Delta_{n,n+1} = [np_\tau, \{n+1\}p_\tau]$ and $\Delta_{n+1,n+2} = (\{n+1\}p_\tau, \{n+2\}p_\tau]$ to find the properties of the bounds given in scheduling theorie. Additional assume $\forall n \in \mathbb{N}_0 : t_n := np_\tau + t'$, then the right-continuous event bound function has the following properties:

$$
\begin{aligned}
\forall t \in \Delta_{n-1,n}: \quad \overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \left\lfloor \frac{t}{p_\tau} + 1 \right\rfloor \\
&= \lim_{t \to np_\tau^-} \left\lfloor \frac{t}{p_\tau} + 1 \right\rfloor \\
&= \left\lfloor \frac{np_\tau}{p_\tau} + 1 \right\rfloor = n+1 \\
\forall t \in \Delta_{n,n+1}: \quad \overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \lim_{t \to np_\tau^+} \left\lfloor \frac{t}{p_\tau} + 1 \right\rfloor \\
&= \left\lfloor \frac{np_\tau}{p_\tau} + 1 \right\rfloor = n+1 \\
\overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \lim_{t \to (n+1)p_\tau^-} \left\lfloor \frac{t}{p_\tau} + 1 \right\rfloor \\
&= \left\lfloor \frac{np_\tau}{p_\tau} + 1 \right\rfloor = n+1 \\
\forall t \in \Delta_{n+1,n+2}: \quad \overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \lim_{t \to (n+1)p_\tau^+} \left\lfloor \frac{t}{p_\tau} + 1 \right\rfloor \\
&= \left\lfloor \frac{(n+1)p_\tau}{p_\tau} + 1 \right\rfloor = n+2
\end{aligned}
$$

The properties of the left-continous event bound are given by

$$
\begin{aligned}
\forall t \in \Delta_{n-1,n}: \quad \overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \left\lceil \frac{t}{p_\tau} \right\rceil = \lim_{t \to np_\tau^-} \left\lceil \frac{t}{p_\tau} \right\rceil \\
&= \left\lceil \frac{np_\tau}{p_\tau} \right\rceil = n \\
\forall t \in \Delta_{n,n+1}: \quad \overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \lim_{t \to np_\tau^+} \left\lceil \frac{t}{p_\tau} \right\rceil \\
&= \left\lceil \frac{np_\tau}{p_\tau} \right\rceil = n \\
\overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \lim_{t \to (n+1)p_\tau^-} \left\lceil \frac{t}{p_\tau} \right\rceil \\
&= \left\lceil \frac{np_\tau}{p_\tau} \right\rceil = n+1 \\
\forall t \in \Delta_{n+1,n+2}: \quad \overset{\bullet\circ}{\mathbb{E}}_\tau(t) &= \lim_{t \to (n+1)p_\tau^+} \left\lceil \frac{t}{p_\tau} \right\rceil \\
&= \left\lceil \frac{(n+1)p_\tau}{p_\tau} \right\rceil = n+1
\end{aligned}
$$

Both functions are inequal in all time points $t := np_\tau$. The event bound considered for only one task can easily be extended to the whole task set: Because $\overset{\bullet\circ}{\mathbb{E}}_\Gamma(t) = \sum_{\tau \in \Gamma} \overset{\bullet\circ}{\mathbb{E}}_\tau(t)$ and $\overset{\circ\bullet}{\mathbb{E}}_\Gamma(t) = \sum_{\tau \in \Gamma} \overset{\circ\bullet}{\mathbb{E}}_\tau(t)$ the inequation $\overset{\circ\bullet}{\mathbb{E}}_\Gamma(t) \not\equiv \overset{\bullet\circ}{\mathbb{E}}_\Gamma(t)$ is true. □

To conclude, the right-continuous event bound and the left-continuous event bound differ in all-time points $t_n = np_\tau$.

4.2 Problem formulation

Is it necessary two use two different functions? The goal of this work is to find a function which is right-continuous in all $t_n = np_\tau$ except the last one which should be left-continuous. Therefore, the utilization test and the response time analysis should use the same function except at the end of the considered timing interval. Remember, the demand bound test evaluates all event requests until the hyper-period $\mathcal{P}$, and the response-time analysis counts all events until the result of the last iteration. In contrast the response time analysis, only the last time point must necessarily be considered and should be left-continuous. The left- and the right-continuous event bounds only differ in their left and right-hand limits of the investigated timing interval. The goal of this work is to find a function which is right-continuous in all $t_n = np_\tau$ except the last one which should be left-continuous. Therefore, both algorithms should use the same function except at the end of the considered timing interval. If we extend the event bound function $\mathbb{E}_\tau : \mathbb{R}^2 \to \mathbb{R}$ we can specify a bound time interval $\Delta_{a,b} := [t_a, t_b) = [a, b)$ which restricts the time in which the event bound counts. If we integrate the hyper-period as bounding restriction to the demand bound function, it is possible to formulate a general unified event bound. Let us discuss this idea in more general:

Problem 1 (Unified event bound function or unified event bound, ueb) *Investigate if a function with the following properties exist: Assume the time interval* $\Delta_{a,b} = [t_a, t_b)$ *and each time point is given by* $\forall n \in \mathbb{N}_0 : t_n := np_\tau + t'$. *In the*

interval $\Delta_{a,b}$ a unified event bound function $\mathbb{E} : \Gamma \times t^2 \to \mathbb{N}$ of any task must fulfill the following properties:

$$t' = 0: \quad \lim_{t' \to 0} \mathbb{E}_\tau(t, \Delta_{a,b}) = n + 1 \tag{9}$$

$$0 < t' < p_\tau: \quad \mathbb{E}_\tau(t, \Delta_{a,b}) = n + 1 \tag{10}$$

$$t' = p_\tau: \quad \lim_{t' \to p_\tau} \mathbb{E}_\tau(t, \Delta_{a,b}) = n + 2 \tag{11}$$

$$t' = p_\tau \wedge t_n = t_b: \quad \lim_{t \to t_b} \mathbb{E}_\tau(t, \Delta_{a,b}) = \lim_{t' \to p_\tau} \mathbb{E}_\tau(t, \Delta_{a,b}) \tag{12}$$

Such a function is equivalent to the right-continuous event bound except at $t = t_b$. Note that for the response time analysis only this point in time is relevant and it is not necessary that all other points of the function are left-continuous. Therefore this function can be used for bound tests tests as well as for response time analysis. A function with these properties are postulated in [12,43]. Both papers accepted an over-approximation by using the right-continuous request bound. These properties follow direct from Lemma 1.

Problem 2 (Postulated demand bound test) *Assume the existence of a unified event bound function. Then the demand bound test in the periodic event model can be written as:*

$$\mathbb{D}_\Gamma(t, \mathcal{P}) \leq t \tag{13}$$

$$\sum_{\tau \in \Gamma} \mathbb{R}_\tau(t - d_\tau, \mathcal{P}) \leq t \tag{14}$$

$$\sum_{\tau \in \Gamma} \mathbb{E}_\tau(t - d_\tau, \mathcal{P}) c_\tau^+ \leq t \tag{15}$$

Problem 3 (Postulated Response Time Analysis) *If a unified event bound function exists, the response time in the periodic event model can be written as:*

$$r_{\tau,n}^+ := c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \mathbb{E}_{\tau'}(r_{\tau,n-1}^+, r_{\tau,n-1}^+) c_{\tau'}^+ \tag{16}$$

$$c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \mathbb{E}_{\tau'}(t, t) c_{\tau'}^+ - t = 0 \tag{17}$$

Assume the following definition for interfering tasks: $\tau' \in \tau \cup \overline{\Gamma}_\tau$, then the response time analysis can be reformulated as the well known fixed-point iteration. In other words, the response time analysis will become a root-finding problem:

$$\sum_{\tau' \in \tau \cup \overline{\Gamma}_\tau} \mathbb{E}_{\tau'}(t, t) c_{\tau'}^+ - t = 0 \tag{18}$$

The idea to integrate the request of the considered task and all higher priority tasks in just one function simplifies the mathematical framework. As we will later see in this paper, the concept of a unified event bound allows the integration of different task models developed during the past decades to only one analysis approach. This opportunity opens a way to integrate a lot of already done work in bound tests tests to the busy-window approach and vice versa. If the unified event bound function exists for event streams as well, the new framework is responsible for each event model published during the last decades.

4.2.1 Goals and organization

The goal of this work is finding a unified event bound and investigating its mathematical properties. As shown in the previous section, the right and left-continuous event bound are only different in their limits at the integer times $t_n = n \cdot p_\tau$. If we assume this as a limited value problem, calculus should be used to solve it. Digital signal theory and theoretical physics already handle discrete events or objects in a continuous environment. Therefore these methods are adapted to real-time analysis. Digital signal processing describes discrete signals by a series of Dirac pulses. This idea can be used to count events, as well. While digital signal processing builds upon a rich mathematical framework, these methods became applicable to real-time systems. This mathematical framework has one significant advantage compared with approaches in related work: Computation is not limited to bound functions. By using Dirac deltas to count events, it is possible to apply algebraic operations directly on events before computing the bound function. This advantage allows constructing constructive interference bounds to model all kind of different real-time schedulers. Therefore, problems in the real-time analysis could be expressed more simply and expressively than in the established work. As we will see later in the paper, the response time of static and dynamic scheduling is computed by only one expression. This new method is so powerful that other priority schemes are described easy in the same way.

The paper is organized as follows: First, we derive a unified event bound function using methods from calculus and distribution theory. Second, we will show how hierarchical event streams can be easily described and computed by using the Dirac delta function. Based on the idea, we develop a unified real-time scheduling theory considering static and dynamic priorities in one holistic approach for bound tests and response time analysis as well. For the first time, we derive both analysis techniques from only one axiom, the average load of a processor. We will then prove past results just by using the new theory. Applying the new theory to past results shows how the unified theory gives an algebraic toolbox for proofs. Now it is not necessary anymore to do any geometric relations on task and job requests in time-based Gantt diagrams as widely done in related work. The new approach is more straightforward and leads to accessible computational models. As a special treat, we can develop a tighter response time analysis as given in related work for dynamic scheduling at the end by just adding the same assumption to dynamic scheduling as already done to model

task with arbitrary deadlines already done in static scheduling. In the end, we will compute the response times of some interesting tasks set in static, dynamic and hierarchical[5] scheduling. The paper ends with an Appendix A concluding the used mathematical symbols and explaining special notations borough from theoretical physics. An additional Appendix B discusses a complete example calculated by a computer algebra system (CAS).

5 The unified event bound function

During the next section, we develop a strict formal view to events as known in signal theory. The idea is to express all needed mathematical properties in the model implicitly without any informal or hidden assumptions. First, we introduce events, and then we show how they can be count in an alternative way compared to the floor and ceil operation. We discuss the mathematical properties and will show how the new method is related to previous work.

5.1 A mathematical view on events and tasks

In real-time systems analysis or scheduling theory, events and jobs introduced semi-formal. Tasks or better jobs were often given as geometrical objects such as rectangles in Gantt charts. Then the length of the rectangle models execution demand of the job and the place of the rectangle determines by its position in time. The hight of the rectangles does not matter and is most often given to 1 as seen in Fig. 2a. The goal of the following section is to formalize release times and time durations appreciatively. Therefore we transform informal geometric proofs to analytical descriptions which are computed algebraically.

5.1.1 Modeling jobs

In each computer system, a computational activity has a duration or in other words, an execution time. The time between the release of a job and its non-preempted execution end starts at a defined point in time t_a and ends later at a second point in time t_b. If the job is not interrupted by any other activity this time is called the worst-case execution time c^+. However, if we assume independent tasks on a unique processor, we can concentrate on c^+. Calling t_a the request time, each job of a task ends after c^+ if no other job interrupts the execution. Therefore the job finishes at $t_b = t_a + c^+$. Figure 2a. shows such a simple behaviour as it is described in most of the previous work by a Gantt–Chart. Therefore, during the execution of a job, the processor is busy and has a utilization of one. In contrast to related work, we first look for an algebraic formulation of this behaviour. Formally the geometric Gantt–Chart description of a job can be replaced by a composition of Heaviside functions.

Definition 9 (*Heaviside function*) Assume $s \in [0, 1]$. The Heaviside or step function $\mathbb{H} : \mathbb{R} \to \{0, 1\}$ is defined[6] as

$$\mathbb{H}(t) = \begin{cases} 0 & t < 0 \\ s & t = 0 \\ 1 & t > 0 \end{cases} \tag{19}$$

Based on this definition, it is easy to introduce the concept of the Dirac delta function or shortly the delta function, which becomes our base to define events formally:

Definition 10 (*Dirac delta*) The Dirac delta function $\delta(t)$ is given by:

$$\mathbb{H}(t) = \int_{-\infty}^{t} \delta(t')\,\mathrm{d}t' \tag{20}$$

This equation does not define a function in a traditional, well-known way. Therefore it is correctly called a distribution. It was first introduced by Paul Dirac in the early 1930s and is a well established mathematical tool in theoretical physics and signal theory [15]. As we will see later, the idea of Paul Dirac can be applied to find and define the unified event bound. It is very important to have in mind that $\delta(t) = 0$ for all $t \neq 0$ which directly follows from the definition.

Let us next consider how any job of task τ with execution time c_τ^+ requested at time $t_{\epsilon'}$ can be modeled. Let us first assume that all jobs has the same execution demand. Therefore we call the task homogenous.

Lemma 2 (Dirac job) *A job requested at time $t_{\epsilon'}$ is modeled by*

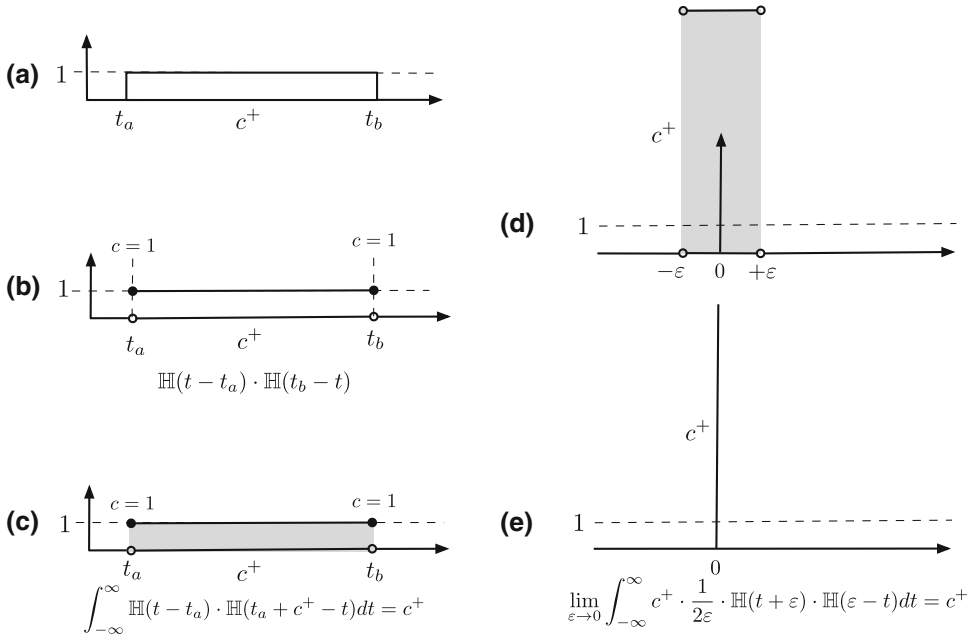
$$c_\tau^+(t_{\epsilon'}) = \int_{-\infty}^{\infty} \delta(t' - t_{\epsilon'}) \cdot c_\tau^+\,\mathrm{d}t' \tag{21}$$

Proof A non-preemptive real-time task instance or job needs two Heaviside functions for its algebraic description: one to represent the request $\mathbb{H}(t - t_a)$ and one to model the completion of the task $\mathbb{H}(t_b - t)$. Consider Fig. 2b. Multiplying both functions builds a rectangle of hight one defined by the execution function $c : \mathbb{R} \to \{0, 1\}$:

[5] In this work hierarchical scheduling means a mixed scheduling policy where any dynamic or static scheduler may embed a scheduler of any lower hierarchy. Therefore we follow the definition of [31] or [50]

[6] Note that different definitions of the Heaviside function exist. The above definition supports the requirements needed in this work best.

Fig. 2 Algebraic task modeling



$$
\begin{aligned}
c_\tau(t,t_a) &= \mathbb{H}(t-t_a)\cdot\mathbb{H}(t_b-t)\\
&= \mathbb{H}(t-t_a)\cdot\mathbb{H}(t_a+c_\tau^+-t)
\end{aligned}
\tag{22}
$$

Alternativ it is possible to use

$$
\begin{aligned}
c_\tau(t,t_a) &= \mathbb{H}(t-t_a)-\mathbb{H}(t-t_b)\\
&= \mathbb{H}(t-t_a)-\mathbb{H}(t-[t_a+c_\tau^+])
\end{aligned}
\tag{23}
$$

It is important to note that such a description does not consider preemption, and therefore, it does not support the bounded execution model completely. As a consequence, it is necessary to model the behaviour of interfering computational loads such as interrupts and higher priority jobs explicitly. The idea of the following is to describe the occurrence frequency of jobs and their requested load concerning the available computation time in a given time interval $[t_a, t]$. Let us first rewrite the equation for the computational load without changing anything:[7]

$$
c_\tau(t,t_a) = \int_{-\infty}^{t}\mathbb{H}(t'-t_a)\cdot\mathbb{H}(t_a+c_\tau^+-t')\,\mathrm{d}t' \tag{24}
$$

The complete non-preempted execution starting at t_a is given by

$$
c_\tau^+(t_a) = \int_{-\infty}^{\infty}\mathbb{H}(t'-t_a)\cdot\mathbb{H}(t_a+c_\tau^+-t')\,\mathrm{d}t' \tag{25}
$$

[7] The integral looks a little bit oversized because both Heaviside functions return only 1. However, introducing the integral is crucial as we will see later.

The release of each job needs a context switch at the beginning and end of execution, and some time ε this context switch. It is common sense in scheduling theory not to consider this time.[8] However, to catch the value limitation problem, we first introduce this overhead, and later it will be removed mathematically:

$$
\begin{aligned}
c_\tau^+(t_a) = &\int_{-\infty}^{\infty}\frac{1}{2\varepsilon}\mathbb{H}(t'-(t_a+\varepsilon))\\
&\cdot\mathbb{H}((t_a+\varepsilon)+c_\tau^+-t')\,\mathrm{d}t'
\end{aligned}
\tag{26}
$$

For our first event ϵ' we are not interested when it starts so let us move it to the origin $t_a = 0$:

$$
c_\tau^+(0) = \int_{-\infty}^{\infty}\frac{1}{2\varepsilon}\mathbb{H}(t'-\varepsilon)\cdot\mathbb{H}(\varepsilon+c_\tau^+-t')\,\mathrm{d}t' \tag{27}
$$

Let us now modify Eq. 27 by describing the computational load not by its horizontal time:

$$
c_\tau^+(0) = \int_{-\infty}^{\infty}\frac{c_\tau^+}{2\varepsilon}\cdot\mathbb{H}(t'-\varepsilon)\cdot\mathbb{H}(\varepsilon-t')\,\mathrm{d}t' \tag{28}
$$

In real-time analysis, we are interested only in the load given by the real-time tasks themselves. Such an assumption is permissible because the job's execution time is much longer than the interruption time by the operating system, and it

[8] If scheduling overhead should be modelled assume a separate task described the operating system.

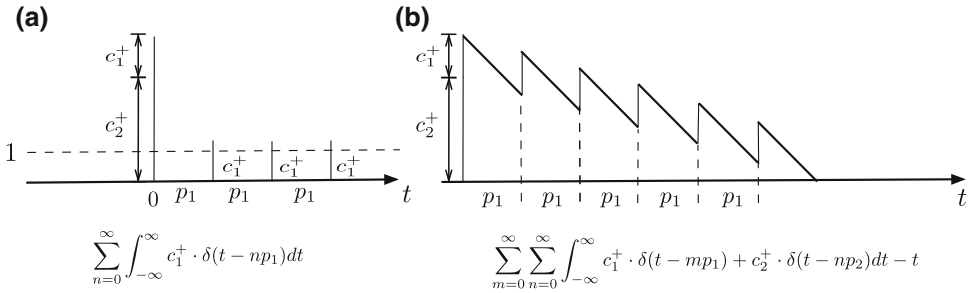


Fig. 3 Modelling preemptive jobs by applying Eq. 31

should be ignored. Therefore we look at what happens if the operating system overhead approaches 0. Mathematically the request span can be simplified under the assumption that $\int_{-\infty}^{\infty} \frac{1}{2\varepsilon}\mathbb{H}(t-\varepsilon)\cdot\mathbb{H}(\varepsilon-t)\,\mathrm{d}t = 1$. The obvious solution to eliminate the time 2ε in the Eq. 28 by setting $\varepsilon = 0$ does not work in general. If we now assume a Heaviside function with $s = 0$ then $\mathbb{H}(t-\varepsilon)\mathbb{H}(\varepsilon-t) = 0$ and not 1. Therefore, we set of ε in a way, that all possible Heaviside functions $s \in [0, 1]$ will be supported as well. Mathematically we apply limit value analysis to the problem: If the term addressed by the integral is divided by 2ε and $\varepsilon \to 0$ we get

$$\lim_{\varepsilon\to 0}\int_{-\infty}^{\infty}\frac{1}{2\varepsilon}\cdot\mathbb{H}(t-\varepsilon)\cdot\mathbb{H}(\varepsilon-t)\,\mathrm{d}t = \int_{-\infty}^{\infty}\delta(0)\,\mathrm{d}t' = 1 \quad (29)$$

Assume the substitution $\delta(0) = \lim_{\varepsilon\to 0}\frac{1}{2\varepsilon}\cdot\mathbb{H}(t-\varepsilon)\cdot\mathbb{H}(\varepsilon-t)$ to rewrite Eq. 28. Consider Fig. 2c. for illustration. Therefore,

$$c_\tau^+(0) = \int_{-\infty}^{\infty}\delta(0)\cdot c_\tau^+\,\mathrm{d}t' \quad (30)$$

And if we like to consider a job requested at time $t_{\epsilon'}$:

$$c_\tau^+(t_{\epsilon'}) = \int_{-\infty}^{\infty}\delta(t'-t_{\epsilon'})\cdot c_\tau^+\,\mathrm{d}t' \quad (31)$$

□

5.1.2 Events as Dirac delta

Let us now apply the delta function by defining events as needed in real-time systems analysis in a strictly formal way:

Definition 11 (*Event*) An event $\epsilon' : \mathbb{R} \to [0, 1]$ is a request at a point in time $t_{\epsilon'} \in \mathbb{R}$ with infinitely short time span:

$$\epsilon'(t_{\epsilon'}) = \begin{cases} \int_{-\infty}^{t}\delta(t'-t_{\epsilon'})\,\mathrm{d}t' = 1 & t = t_{\epsilon'} \\ \delta(t-t_{\epsilon'}) = 0 & t \neq t_{\epsilon'} \end{cases} \quad (32)$$

The time point $t_{\epsilon'}$ calls the request time of the event.

In other words, an event ist a timeless state change in any system. Computing only the area bounded by a given Heaviside function does not allow to consider preemption as needed in the bounded execution model. Multiplying a Dirac delta with any given WCET results in a peak with the amplitude of the execution time at the request time of the event, as shown in Fig. 3a by alieng equation . Running overtime t the value of these peaks is reduced exactly by t in the interval t. Because the model considers the release time of events, we can add a peak of execution time at any time an interfering job of higher priority interrupts the execution of the considered job. Consider Fig. 3 which illustrates the idea. At time $t = 0$ task τ_1 and τ_2 are requested. After the specified period p_1 task τ_1 is requested again. Figure 3b. shows the behaviour of the resulting function. Note, that such a saw-function is equal to the well-known request bound function of these two tasks subtracting t. Changing the point of view transforms the established fixed-point iteration of the busy window approach to find the roots of the equivalent sawtooth-wave.

5.1.3 Event models

The definition of only one event does not support the modelling of tasks as a sequence of jobs. Therefore a formal description of a series or sequence of events is required to model sequential jobs. Mathematically this is expressed by a series of Dirac deltas called a Dirac comb in the case all events are strictly periodic. However, the general way to describe any sequence of requesting events is to describe event streams. An event stream can be described by a Dirac comb as well:

Definition 12 (*Event density*) A sequence of k events is given by:

$$\text{ш}_\tau(k,t) = \sum_{\epsilon\in\mathcal{E}_\tau}\sum_{n=0}^{k-1}\delta(t-\phi_\epsilon-np_\epsilon) \quad (33)$$

calling $\text{Ш}_\tau(k,t)$ an event stream or event density.[9] Therefore an event sequence[10] ϵ specifying k periodic events ϵ' with offset can be written with the event tuple:

$$\epsilon =< p, \phi >_k \tag{34}$$

Moreover, as a short form notation a set of corresponding event tuples defines the event density formally:

$$\mathcal{E} = \{< p, \phi >_k\} \tag{35}$$

We choose the notation $< a, b >_k$ to distinguish the new approach clearly from the event stream notation.

This definition introduces a new perspective and insight into event streams. A mathematical equation now describes an event stream with precisely defined mathematical properties instead of only writing a weak set of tuples. To describe event densities which model valid event streams, we assume a maximal event density Ш^+_Δ and a minimal event density Ш^-_Δ. Both are event densities which have the mathematical property of sub- or super-additivity. Additionally, it is very easy to bound the number of events. Instead of previous models, the term event density allows specifying a fixed number of events as a sporadic or bursty event stream.

Example 2 (Periodic event model) The periodic event model describes an infinite number of periodic events. Assume $\phi_\epsilon = 0$, therefore $k = \infty$ and the sequence of events is given by

$$\text{Ш}_\tau(\infty, t) = \sum_{n=0}^{\infty} \delta(t - np_\tau) = \sum_{n \in \mathbb{N}_0} \delta(t - np_\tau) \tag{36}$$

Assume a sporadic event which occurs only once. In the event stream model, the definition of the event bound requires to set the period of the given event tuple of the sporadic event to $p_\tau = \infty$. Now the period is zero if an event occurs only once and the sum limits the occurrence:

Example 3 (Sporadic event) A event which is sporadic and which occurs only at $t_\epsilon = 10\,\text{ms}$ is described by

$$\begin{aligned} \mathcal{E} &= \{< 0, 10 >_1 \text{ ms}\} \\ &= \{< 0, 10 > \text{ ms}\} \quad \text{in contrast to} \quad \mathcal{E} = \left\{ \binom{\infty}{10} \text{ ms} \right\} \end{aligned} \tag{37}$$

as originally defined by [23].

[9] This work introduces the term 'event density'. As we will see later, this is a more intuitive term than the name event stream as used in previous work.

[10] We distinguish between the event density as a sum of Dirac deltas and the event tuple describing the parameters of the event density.

Example 4 (Periodic event model with jitter) First assume the established periodic event model with jitter:

$$\begin{aligned} \mathcal{E}_{Jitter} &= \{< 0, 0 >_1, < p, p - 2j >_\infty\} \\ &= \{< 0, 0 >, < p, p - 2j >\} \end{aligned} \tag{38}$$

Now consider we only like to describe four events in this model:

$$\mathcal{E}_{Jitter} = \{< 0, 0 >_1, < p, p - 2j >_3\} \tag{39}$$

Such a description is natural, easier to understand, and more potent than the original form.

5.1.4 To count or not to count

After defining the event density, we have to consider how to count the events. As we have seen in Lemma 2, the execution time of a job is computed by integrating a couple of Dirac deltas. Therefore, we will find the number of events by integrating over a series of Dirac deltas or events, called event density. However, the integral gives us the freedom two sum over event densities in any given time interval. As we will see, this is a significant advantage compared to the counting of events in related work.

5.1.5 Counting by integrating dirac deltas

To compute the execution demand of a processor, events and a series of events must be counted during a given time. This number of events then is multiplied by the specified execution time of the task. By changing the event model to Dirac deltas, we have to count the number of deltas in a given timing interval. Integrating the series of Dirac deltas results in the number of events given in the time bounded by the limits of the integral:

Lemma 3 (Finite event bound) *Assume any time interval* $\Delta_{a,b} := [t_a, t_b] \in \mathbb{R}$. *The number of events then can be counted by a function* $\mathbb{E} : \Gamma \times \mathbb{R} \times \mathbb{N} \to \mathbb{N}$

$$\begin{aligned} \mathbb{E}_\tau(\Delta_{a,b})_k &= \int_{\Delta_{a,b}} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon)\, \mathrm{d}t' \\ &= \int_{t_a}^{t_b} w_\tau(k, t')\, \mathrm{d}t' \end{aligned} \tag{40}$$

Also, in the particular case of the periodic event model with countless events, this simplifies to:

$$\mathbb{E}_\tau(\Delta_{a,b}) = \int_{t_a}^{t_b} \sum_{n \in \mathbb{N}_0} \delta(t' - np_\tau)\, \mathrm{d}t' = \left\lfloor \frac{\Delta_{a,b}}{p_\tau} + 1 \right\rfloor \tag{41}$$

Proof We have to sum all events of an event density:

$$\sum_{\epsilon \in \mathcal{E}_\tau} \epsilon(t_\epsilon) \tag{42}$$

According to the definition of an event $\epsilon(t_\epsilon) == \int_{-\infty}^{\infty} \delta(t' - t_\epsilon)t'$, we get for any event density in $\Delta_{a,b}$

$$\mathbb{E}_\tau(\Delta_{a,b})_k = \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \int_{t_a}^{t_b} \delta(t' - \phi_\epsilon - np_\epsilon)\,\mathrm{d}t' \tag{43}$$

By definition the number of event tuples is limited and the series given by equation 43 converges absolute for $k \in [0, \infty)$, because of the bound $\Delta_{a,b}$ and $n \in \mathbb{N}$. In the case of a finite number of events, the convergence of the sum is trivial. Therefore, the integral and the sum can be switched:

$$\mathbb{E}_\tau(\Delta_{a,b}) = \int_{t_a}^{t_b} \sum_{n \in \mathbb{N}_0} \delta(t' - np_\tau)\,\mathrm{d}t' = \left\lfloor \frac{\Delta_{a,b}}{p_\tau} + 1 \right\rfloor \tag{44}$$

Note, that the first assumption does not holt if $\Delta_{a,b} \in (-\infty, \infty)$. However, as we will see later, the sum or event density is always limited. □

By definition of the unified event bound, we do not want to count events at t_b. However, by definition in calculus, the Riemann integral is bounded by the interval $[t_a, t_b]$ and this is not the postulated interval $\Delta_a^b \in [t_a, t_b)$. Transforming the limits of the integral in a right-open interval can be done by masking the desired interval with the help of Heaviside functions. In this case, two variants of the infinite set of Heaviside functions as defined in Definition 9 are needed:

Definition 13 (*Upper Heaviside function*) Assume the general Heaviside function with $s \in [0, 1]$ and let $s = 1$. Then the upper Heaviside function $\overline{\mathbb{H}} : \mathbb{R} \to \{0, 1\}$ is

$$\overline{\mathbb{H}}(t) = \begin{cases} 0 & t < 0 \\ 1 & t \geq 0 \end{cases} \tag{45}$$

Definition 14 (*Lower Heaviside function*) Assume the general Heaviside function with $s \in [0, 1]$ and let $s = 0$. Then the lower Heaviside function $\underline{\mathbb{H}} : \mathbb{R} \to \{0, 1\}$ is

$$\underline{\mathbb{H}}(t) = \begin{cases} 0 & t \leq 0 \\ 1 & t > 0 \end{cases} \tag{46}$$

By applying Definitions 13 and 14 to Lemma 3 we can formulate the unified event bound as illustrated in Fig. 4:

Theorem 1 (Unified event bound function, ueb) *Assume* $\mathbb{E} : \Gamma \times \mathbb{R}^2 \times \mathbb{N} \to \mathbb{N}$, *then the number of events in any bounded interval* $\Delta_{a,b} = [t_a, t_b)$ *can be counted by*

$$\begin{aligned} \mathbb{E}_\tau(t, \Delta_{a,b})_k &= \int_{-\infty}^{t} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \\ &\quad \cdot \overline{\mathbb{H}}\left(t' - t_a\right) \cdot \underline{\mathbb{H}}\left(t_b - t'\right)\,\mathrm{d}t' \qquad (47) \\ &= \int_{t_a}^{t} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \\ &\quad \cdot \underline{\mathbb{H}}\left(t_b - t'\right)\,\mathrm{d}t' \qquad (48) \end{aligned}$$

in the special case of a periodic event model this becomes

$$\begin{aligned} \mathbb{E}_\tau(t, \Delta_{a,b})_k &= \int_{-\infty}^{t} \sum_{n \in \mathbb{N}_0} \delta(t' - np_\tau) \\ &\quad \cdot \overline{\mathbb{H}}\left(t' - t_a\right) \cdot \underline{\mathbb{H}}\left(t_b - t'\right)\,\mathrm{d}t' \qquad (49) \\ &= \int_{t_a}^{t} \sum_{n \in \mathbb{N}_0} \delta(t' - np_\tau) \cdot \underline{\mathbb{H}}\left(t_b - t'\right)\,\mathrm{d}t' \qquad (50) \end{aligned}$$

Proof Assume we count the events in a bounded interval $[t_a, t_b]$. Instead of the infinite interval given in Eq. 40, we bound the integral by its limits:

$$\mathbb{E}_\tau([t_a, , t_b])_k = \int_{t_a}^{t_b} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon)\,\mathrm{d}t' \tag{51}$$

The limits of the integration include t_a and t_b by definition. While $\overline{\mathbb{H}}\left(t' - t_a\right) \cdot \overline{\mathbb{H}}\left(t_b - t'\right) = 1$ only if $t_a \leq t \leq t_b$ and 0 in all other cases, equation (51) can be rewritten as:

$$\begin{aligned} \mathbb{E}_\tau([t_a, t_b])_k &= \int_{t_a}^{t_b} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon)\,\mathrm{d}t' \qquad (52) \\ &= \int_{-\infty}^{\infty} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \\ &\quad \cdot \overline{\mathbb{H}}\left(t' - t_a\right) \cdot \overline{\mathbb{H}}\left(t_b - t'\right)\,\mathrm{d}t' \qquad (53) \end{aligned}$$

Changing the term $\overline{\mathbb{H}}\left(t_b - t'\right)$ to $\underline{\mathbb{H}}\left(t_b - t'\right)$ excludes t_b from the bound.[11] Therefore, the integration over $\Delta_{a,b} \in [t_a, t_b)$

[11] This is an excellent mathematical trick to bound integrals to open intervals borrowed from physics.

can be formulated as

$$\mathbb{E}_\tau([t_a, , t_b))_k = \int_{-\infty}^{\infty} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \cdot \overline{\mathbb{H}}\left(t' - t_a\right) \cdot \underline{\mathbb{H}}\left(t_b - t'\right) \, \mathrm{d}t' \tag{54}$$

We observe that this integral is not only bounded by $\Delta_{a,b}$. Assume $t_a \leq t < t_b$, then this function is also bounded by t, and therefore we can write[12]

$$\mathbb{E}_\tau(t, \Delta_{a,b})_k = \int_{-\infty}^{t} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \cdot \overline{\mathbb{H}}\left(t' - t_a\right) \cdot \underline{\mathbb{H}}\left(t_b - t'\right) \, \mathrm{d}t' \tag{55}$$

$$= \int_{t_a}^{t} \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \cdot \underline{\mathbb{H}}\left(t_b - t'\right) \, \mathrm{d}t' \tag{56}$$

The proof for the periodic or sporadic model is obvious. □

In worst case response time analysis and the demand bound test, the starting time of the analysis interval is implicitly set to $t_a = 0$ by definition. Therefore, $\Delta_{0,b} = [0, t_b)$. Then the unified event bound function can be written as

$$\mathbb{E}_\tau(t, t_b)_k = \int_0^t \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \cdot \underline{\mathbb{H}}\left(t_b - t'\right) \, \mathrm{d}t' \tag{57}$$

Compare Eq. 57 with Eq. 41 given by Lemma 3 which shows the limitation of the scheduling theory: In contrast to previous work, the unified event bound allows computing the number of events in any time interval. Therefore the computation of the bound can be moved to any time point $t \in \mathbb{R}$, and the unified event bound is invariant in time. However, Theorem 1 has additional properties useful in real-time scheduling analysis: Defining the event bound by integrating over Dirac pulses and limiting this integral by two different Heaviside functions, the upper and lower Heaviside function, we find several and different bounds if we combine different descriptions for integral limits. This results in four different cases:

i The bound of the number of events k.
ii The lower timing bound t_a defined by, the lower Heaviside mask or, the lower limit of the integral.
iii The timing bound t given by the limitation of the Dirac comb or the upper limit of the integral.
iv The above timing bound t_b as defined by the upper Heaviside mask.

The first result of this paper shows that previous work defined different event bound functions because not considering the limits of time intervals like in calculus. We have proven that a unified function has to consider the limits of a well-defined integration problem as we can see in Fig. 4. Additionally, we know from distribution theory, that the Dirac delta is the derive of the Heaviside function [15]. As a consequence, it is obvious to call an event stream an event density: The event count in the real-time analysis is equal to an integral over a dense series of Dirac deltas.

Definition 15 (*Heaviside mask*) The pair of Heaviside functions limits the integration interval by masking bounds:

$$\mathbb{M}(t, \Delta_{a,b}) := \mathbb{H}(t - t_a) \cdot \mathbb{H}(t_b - t) \tag{58}$$

With $\mathbb{H}(t - t_a)$ the left or early mask and $\mathbb{H}(t_b - t)$ the right or late mask. Note, that both Heaviside functions can be upper or lower Heaviside functions. Therefore four different masks exist: $\mathbb{M}(t, \Delta_{\overline{a}}^{\overline{b}})$, $\mathbb{M}(t, \Delta_{\overline{a}}^{\underline{b}})$, $\mathbb{M}(t, \Delta_{\underline{a}}^{\overline{b}})$ and $\mathbb{M}(t, \Delta_{\underline{a}}^{\underline{b}})$.

5.1.6 Traditional unified event bound

The previous section concludes that the unified event bound is a problem of finding the correct limitations or bounds to count the events. Let us investigate whether it is possible to reformulate the standard tests as given by equation 7 in a unified way. In this section, we study if it is possible to express the floor and ceil function by the unified approach as well.

Lemma 4 (Right-continuous event bound) *If $t \in \mathbb{R}$, the right-continuous event bound function is equal to a sum of Heaviside functions bounded by $\overline{\mathbb{H}}(0)$ and $\overline{\mathbb{H}}(\infty)$ which is equal to the limits of an integral:*

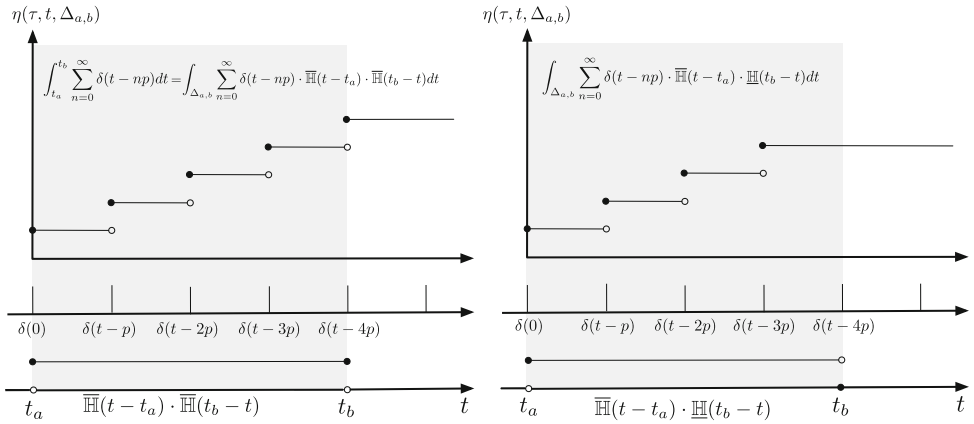
$$\overset{\bullet\!\!\circ}{\mathbb{E}}_\tau(t) = \left\lfloor \frac{t}{p_\tau} + 1 \right\rfloor = \int_0^t \sum_{n \in \mathbb{N}_0} \delta(t' - np_\tau) \, \mathrm{d}t' \tag{59}$$

Proof Assume any time point $t = np + t''$ in the periodic event model. The event bound only changes in the points $t_n = np$. Therefore, we have only to check what happens when the time t approaches np from the left and the right site:

$$\overset{\bullet\!\!\circ}{\mathbb{E}}_\tau(t) = \int_0^t \sum_{n \in \mathbb{N}_0} \delta(t' - np_\tau) \, \mathrm{d}t' = \lim_{t'' \to 0} \int_0^{np_\tau + t''} \sum_{n \in \mathbb{N}_0} \delta(t' - np_\tau) \, \mathrm{d}t' \tag{60}$$

[12] The integral $\int_a^t t' \, \mathrm{d}t'$ is defined on the interval $[a, t]$. Therefore the above simplification holds.

Fig. 4 To count or not to count



$$= \lim_{t''\to 0} \int_{-\infty}^{\infty} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau) \cdot \overline{\mathbb{H}}\left(t'-0\right) \cdot \overline{\mathbb{H}}\left(np_\tau + t'' - t'\right) \mathrm{d}t' \tag{61}$$

$$= \int_{-\infty}^{\infty} \sum_{0}^{n} \delta(t'-np_\tau) \cdot \overline{\mathbb{H}}\left(t'\right) \cdot \overline{\mathbb{H}}\left(np_\tau - t'\right) \mathrm{d}t' \tag{62}$$

$$= \sum_{0}^{n} \int_{-\infty}^{\infty} \delta(t'-np_\tau) \cdot \overline{\mathbb{H}}\left(t'\right) \cdot \overline{\mathbb{H}}\left(np_\tau - t'\right) \mathrm{d}t' \tag{63}$$

$$= (n+1) \int_{-\infty}^{\infty} \delta(0)\, \mathrm{d}t' = n+1 \tag{64}$$

Now, we apply Lemma 3 at the last step.[13] Let $t'' \in (0,\, p_\tau)$, then the number of events is computed by

$$\overset{\bullet\circ}{\mathbb{E}}_\tau\ (t) = \int_{-\infty}^{\infty} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau) \cdot \overline{\mathbb{H}}\left(t'-0\right) \cdot \overline{\mathbb{H}}\left(t-t'\right) \mathrm{d}t' \tag{65}$$

$$= \int_{0}^{np_\tau+t''} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau)\, \mathrm{d}t' \tag{66}$$

$$= \int_{0}^{np_\tau} \sum_{0}^{n} \delta(np_\tau - np_\tau)\, \mathrm{d}t' + \int_{np_\tau}^{t''} \sum_{0}^{n} \delta(t'-np_\tau)\, \mathrm{d}t' \tag{67}$$

$$= \int_{-\infty}^{\infty} \sum_{0}^{n} \delta(0)\, \mathrm{d}t' + \int_{0}^{t''} \sum_{0}^{n} \delta(t'')\, \mathrm{d}t' \tag{68}$$

$$= \int_{-\infty}^{\infty} (n+1)\delta(0)\, \mathrm{d}t' + 0 = n+1 \tag{69}$$

[13] Note that the event bound is limited by the Heaviside mask, then convergence is ensured.

Note, that if $t'' \in (0,\, p_\tau)$ no event is counted in this interval. Last we have to prove the equality for $t = np_\tau + p_\tau$:

$$\overset{\bullet\circ}{\mathbb{E}}_\tau\ (t) = \int_{-\infty}^{\infty} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau) \cdot \overline{\mathbb{H}}\left(t'-0\right) \cdot \overline{\mathbb{H}}\left(t-t'\right) \mathrm{d}t' \tag{70}$$

$$= \int_{0}^{t} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau)\, \mathrm{d}t'$$

$$= \lim_{t''\to p_\tau} \int_{0}^{np_\tau+t''} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau)\, \mathrm{d}t' \tag{71}$$

$$= \int_{0}^{np_\tau+p_\tau} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau)\, \mathrm{d}t'$$

$$= \int_{-\infty}^{\infty} \sum_{0}^{n+1} \delta((n+1)p_\tau - np_\tau)\, \mathrm{d}t' \tag{72}$$

$$= \int_{-\infty}^{\infty} (n+2)\delta(0)\, \mathrm{d}t' = n+2 \tag{73}$$

In all three cases the number of counted events is equal to the number of events given in the proof of lemma 1. □

After showing that the new approach can express the right continuous event bound, we check how the left continuous event bound function can be described:

Lemma 5 (Left-continuous event bound function) *Assume t, t' and $t'' \in \mathbb{R}^+$. The left-continous event bound is equal to an integral over Dirac deltas bounded by $\underline{\mathbb{H}}\left(t'-0\right) = \underline{\mathbb{H}}\left(t'\right)$ and $\underline{\mathbb{H}}\left(t-t'\right)$:*

$$\overset{\circ\bullet}{\mathbb{E}}_\tau\ (t) = \left\lceil \frac{t}{p_\tau} \right\rceil = \int_{-\infty}^{\infty} \sum_{n\in\mathbb{N}_0} \delta(t'-np_\tau) \cdot \overline{\mathbb{H}}\left(t'\right)$$

$$\cdot \underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' \tag{74}$$

Proof Assume any time point $t = np_\tau + t''$ in the periodic event model. Again we check how many events are counted at $t = np_\tau, t = np_\tau + t''$ and $t = np_\tau + p_\tau$. If $t = np_\tau$ then $t'' = 0$ and the number of events is given by

$$\overset{\bullet\circ}{\mathbb{E}}_\tau(t) = \int_{-\infty}^{\infty} \sum_{n'\in\mathbb{N}_0} \delta(t'-n'p_\tau)\cdot\overline{\mathbb{H}}\left(t'\right)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' \tag{75}$$

$$= \lim_{t''\to 0}\int_{-\infty}^{\infty} \sum_{n'\in\mathbb{N}_0} \delta(t'-n'p_\tau)\cdot\overline{\mathbb{H}}\left(t'\right)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' \tag{76}$$

$$= \int_{-\infty}^{\infty}\sum_{0}^{n}\delta(0)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' = \int_{-\infty}^{\infty}\sum_{0}^{n-1}\delta(0)\,\mathrm{d}t' \tag{77}$$

$$= \int_{-\infty}^{\infty} n\cdot\delta(0)\,\mathrm{d}t' = n \tag{78}$$

Now let $t'' \in (0, p_\tau)$, then the number of events is computed by

$$\overset{\bullet\circ}{\mathbb{E}}_\tau(t) = \int_{-\infty}^{\infty} \sum_{n'\in\mathbb{N}_0} \delta(t'-n'p_\tau)\cdot\overline{\mathbb{H}}\left(t'-0\right)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' \tag{79}$$

$$= \int_{0}^{np_\tau+t''} \sum_{n'\in\mathbb{N}_0} \delta(t'-n'p_\tau)\cdot\overline{\mathbb{H}}\left(t'\right)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' \tag{80}$$

$$= \int_{0}^{np_\tau}\sum_{0}^{n}\delta(t'-n'p_\tau)\cdot\overline{\mathbb{H}}\left(t'\right)\,\mathrm{d}t' \tag{81}$$

$$= \int_{-\infty}^{\infty}\sum_{0}^{n}\delta(0)\,\mathrm{d}t' = \int_{-\infty}^{\infty}(n+1)\delta(0)\,\mathrm{d}t' = n+1 \tag{82}$$

Last, we have to prove the equality for $t = np_\tau + p_\tau$:

$$\overset{\bullet\circ}{\mathbb{E}}_\tau(t) = \int_{-\infty}^{\infty} \sum_{n'\in\mathbb{N}_0} \delta(t'-n'p_\tau)\cdot\overline{\mathbb{H}}\left(t'-0\right) \cdot\underline{\mathbb{H}}\left(t'-t'\right)\,\mathrm{d}t' \tag{83}$$

$$= \lim_{t''\to p_\tau}\int_{0}^{np_\tau+t''} \sum_{n'\in\mathbb{N}_0} \delta(t'-n'p_\tau)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' \tag{84}$$

$$= \int_{0}^{(n+1)p_\tau} \sum_{n'\in\mathbb{N}_0} \delta(t'-n'p_\tau)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' \tag{85}$$

$$= \int_{-\infty}^{\infty}\sum_{0}^{n+1}\delta(0)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t' = \int_{-\infty}^{\infty}\sum_{0}^{n}\delta(0)\,\mathrm{d}t' = n+1 \tag{86}$$

In all three cases the number of counted events is equal to the number of events given in the proof of lemma 1. □

We can conclude from the previous considerations that it should be possible to formulate the unified event bound by using the ceil or floor operators. As the unified event bound has to replace the left-continuous event bound in most timing points, we construct the unified event bound with the floor operator:

Theorem 2 (Unified event bound with floor operator) *Assume the interval $\Delta_{a,b} = [t_a, t_b)$:*

$$\mathbb{E}_\tau(t, \Delta_{a,b}) = \sum_{\epsilon\in\mathcal{E}}\left\lfloor \frac{t-\phi_\epsilon}{p_\epsilon} + \underline{\mathbb{H}}\left(t_b - t\right)\right\rfloor \cdot \underline{\mathbb{H}}\left(t-t_a\right) \tag{87}$$

and in the periodic model

$$\mathbb{E}_\tau(t, \Delta_{a,b}) = \left\lfloor \frac{t}{p_\tau} + \underline{\mathbb{H}}\left(t_b - t\right)\right\rfloor \cdot \underline{\mathbb{H}}\left(t-t_a\right) \tag{88}$$

is equal in the interval, and only in the $\Delta_{a,b} = [t_a, t_b)$ to the unified event bound function given in Eq. 47. Note, that if $t_a < 0$ and $t > t_b$, Eqs. 47 and 87 are not equal.

Proof For simplification and compatibility to related work we prove this only for the periodic event model. If $t_a \le t < t_b$ then $\overline{\mathbb{H}}\left(t-t_a\right) = 1$ and $\underline{\mathbb{H}}\left(t_b - t\right) = 1$. Then

$$\forall t < t_b : \mathbb{E}_\tau(t, \Delta_{a,b}) = \left\lfloor \frac{t}{p_\tau} + \underline{\mathbb{H}}\left(t_b - t\right)\right\rfloor = \left\lfloor \frac{t}{p_\tau} + 1\right\rfloor \tag{89}$$

and if $t \ge t_b$ then $\underline{\mathbb{H}}\left(t_b - t\right) = 0$ and $\underline{\mathbb{H}}\left(t_b - t\right) = 1$

$$\forall t \ge t_b : \mathbb{E}_\tau(t, \Delta_{a,b}) = \left\lfloor \frac{t}{p_\tau} + \underline{\mathbb{H}}\left(t_b - t\right)\right\rfloor = \left\lfloor \frac{t}{p_\tau}\right\rfloor \tag{90}$$

According to Lemma 4, Eq. 59 is equal to Eq. 89 if $t < t_b$ and Eq. 74 is equivalent to 90 if $t \ge t_b$. □

If we want to express the traditional event bound used in the demand bound test, we can write $t' = \Delta_{0,t_b} = [0, t_b)$, and therefore

$$\mathbb{E}_\tau(t, t') = \left\lfloor \frac{t}{p_\tau} + \underline{\mathbb{H}}\left(t' - t\right)\right\rfloor \tag{91}$$

Someone will remark that Theorem 1 is hard to compute numerical. However, Theorem 2 shows that this can easily be done traditionally just by modifying the well-known event bound. Therefore the new approach is a theoretical improvement of the previous related work and extends the theory. It is important to recognize that Eq. 91 does not hold for $t \ge t_b + p$. In this case, both functions are not equal anymore. It should be easily shown that a full compatible unified

Table 1 Relations between traditional [10,32] and uniform theory in the interval $[0, t]$

	Traditional model		Unified model
$\overset{\bullet\!-\!\circ}{\mathbb{E}}_\tau\ (t)$	$\left\lceil \frac{t}{p_\tau} \right\rceil$	=	$\int_{-\infty}^{\infty} \sum_{n\in\mathbb{N}_0} \delta(t-np_\tau)\cdot\overline{\mathbb{H}}\left(t'\right)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t'$
F	$\left\lfloor \frac{t}{p} \right\rfloor$	=	$\int_{-\infty}^{\infty} \sum_{n\in\mathbb{N}_0} \delta(t-np_\tau)\cdot\underline{\mathbb{H}}\left(t'\right)\cdot\underline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t'$
$\overset{\circ\!-\!\bullet}{\mathbb{E}}_\tau\ (t)$	$\left\lfloor \frac{t}{p_\tau}+1 \right\rfloor$	=	$\int_{-\infty}^{t} \sum_{n\in\mathbb{N}_0} \delta(t-np_\tau)\cdot\overline{\mathbb{H}}\left(t'\right)\cdot\overline{\mathbb{H}}\left(t-t'\right)\,\mathrm{d}t'$
$\mathbb{E}_\tau(t, \Delta_a^b)$	$\left\lfloor \frac{t}{p_\tau}+\underline{\mathbb{H}}\left(b-t'\right) \right\rfloor$	=	$\int_{-\infty}^{t} \sum_{n\in\mathbb{N}_0} \delta(t-np_\tau)\cdot\overline{\mathbb{H}}\left(t'-a\right)\cdot\underline{\mathbb{H}}\left(b-t'\right)\,\mathrm{d}t'$

event bound with the floor operation exists. However, the traditional related work can not limit the number of events by a given number of events k. In traditional notation, we always assume $k = \infty$. Table 1 concludes the results.[14]

6 Unified analysis of real-time systems

In this section, we consider how the unified event bound can be used to solve real-time analysis problems. As the new event model is more expressive than the traditional model, we will present some impressive results. For the first time in real-time analysis, it is possible to formulate the bound tests test and the response time analysis only by different expressions of the same mathematical approach in static as in dynamic scheduling as well. For the first time in real-time analysis, it is possible to formulate the bound tests test and the response time analysis only by different expressions of the same mathematical approach in static as in dynamic scheduling as well. Furthermore, it is possible to model additional conditions on task scheduling without modification of the structure of our analysis equation. Therefore it is easy to derivate variants to model bursty or hierarchical event patterns or hierarchical schedulers.[15] The first step in this section is to investigate the relationship between digital signal processing and real-time analysis. Then we will consider some useful definitions and preliminaries to derivate the analysis equations for static and dynamic bound tests and response time analysis from only two axioms. Moreover, in the end, we discuss the results related to previous work.

6.1 A general event model: The event spectrum

Simple event models become complex in bursty events. Different solutions address this problem [2,48]. However, both approaches are not intuitive and require different models to describe the synchronization of events. The two papers solve the problem in different ways but lack to give mathematical or formal approaches to their solutins. Applying now the mathematical toolset developed in Sect. 5 a hierarchical event stream as described by [2] can be derived mathematically. Assume two independent event densities: One event density with a small period and a second one with a much larger one. Both densities together form a new bursty event stream if they are synchronized. The convolution of Dirac combs computes the composition of two event densities. Therefore, synchronization in real-time scheduling can be modelled by

Theorem 3 (Hierarchical event density composition) *Any hierarchical event stream is a composition of two flat event densities and can be computed by the convolution of the two event densities:*

$$\hat{ш}^{k,l}_{\epsilon_o,\epsilon_i} = ш^k_{\mathcal{E}_1} * ш^l_{\mathcal{E}_2} \tag{92}$$

Proof To make the proof easy to follow, we assume $\delta(\tau - t)_\epsilon = \sum_{\epsilon\in\mathcal{E}} \sum_{n=0}^{k-1} \delta(t-\phi_\epsilon - np_\epsilon)$ and $t_n = \phi_{\epsilon_1} + np_{\epsilon_1}$ and $t_m = \phi_{\epsilon_2} + mp_{\epsilon_2}$:

$$ш^k_{\mathcal{E}_1} * ш^l_{\mathcal{E}_2} = \int_{-\infty}^{\infty} ш^k_{\mathcal{E}_1} \cdot ш^l_{\mathcal{E}_2}\,\mathrm{d}\tau \tag{93}$$

$$= \int_{-\infty}^{\infty} \delta(\tau - t_n)_{\epsilon_1} \cdot \delta(t - \tau - t_m)_{\epsilon_2}\,\mathrm{d}\tau \tag{94}$$

$$= \int_{-\infty}^{\infty} \delta(\tau - t_n)_{\epsilon_1} \cdot \delta(t - [\tau + t_m])_{\epsilon_2}\,\mathrm{d}\tau \tag{95}$$

Substitute $\xi = \tau - t_n$ and $d\tau = d\xi$:

$$ш^n_{t,k} * ш^m_{t,l} = \int_{-\infty}^{\infty} \delta(\xi)_{\epsilon_1} \cdot \delta(t - [\xi + t_n + t_m])_{\epsilon_2}\,\mathrm{d}\xi \tag{96}$$

[14] [32] introduced the symbol F to proof optimality.

[15] Such schedulers are called hierarchical in real-time calculus. However, [32] call it mixed schedulers. Therefore term also differs from [11,26,52]. In this paper, we mention a scheduler which schedules all jobs with the same priority according to their dynamic deadlines.

$$= \int_{-\infty}^{\infty} \delta(\xi)_{\epsilon_1} \cdot \delta(t - [t_n + t_m] - \xi)_{\epsilon_2} \, \mathrm{d}\xi \tag{97}$$

For $\xi \neq 0$, the trival solution is $\text{ш}_{t,k}^n * \text{ш}_{t,l}^m = 0$. The only nontrivial solution of the last equation is for $\xi = 0$: We get $\int_{-\infty}^{\infty} \delta(0)_{\epsilon_1} \, \mathrm{d}\xi = 1$ and therefore

$$\int_{-\infty}^{\infty} \delta(0)_{\epsilon_1} \, \mathrm{d}\xi \cdot \delta(t - [t_n + t_m])_{\epsilon_2} = \delta(t - [t_n + t_m])_{\hat{\mathcal{E}}} \tag{98}$$

$$\text{ш}_{t,k}^n * \text{ш}_{t,l}^m = \sum_{n=0}^{k-1} \sum_{m=0}^{l-1} \delta(t - [\phi_{\epsilon_1} + np_{\epsilon_1} + \phi_{\epsilon_2} + mp_{\epsilon_2}]) \tag{99}$$

$$= \sum_{n=0}^{k-1} \sum_{m=0}^{l-1} \delta(t - [\phi_{\epsilon_1} + \phi_{\epsilon_2} + np_{\epsilon_1} + mp_{\epsilon_2}]) \tag{100}$$

In other words, the product of the Dirac delta becomes zero, exactly if $\tau - t_n = 0$ and if $\tau + t_m = t$. Therefore the theorem holds. □

Definition 16 (*Event spectrum*) As a result from the previous theorem the following 3-tuple describes hierarchical and synchronized event densities:

$$\epsilon =< \phi_{\epsilon_o} + \phi_{\epsilon_i}, p_{\epsilon_o}, p_{\epsilon_i} >_{k,l} \tag{101}$$

with the hierarchical event density or event spectrum

$$\hat{\text{ш}}_{\epsilon_o,\epsilon_i}^{k,l} = \sum_{n=0}^{k-1} \sum_{m=0}^{l-1} \delta(t - [\phi_{\epsilon_o} + \phi_{\epsilon_i} + np_{\epsilon_o} + mp_{\epsilon_i}]) \tag{102}$$

The event spectrum is the most general form of an event model. An event spectrum can express all other known event models. According to Lemma 2, the event bound is calculated only by integrating the event spectrum density. Additionally, Theorem 3 gives us the possibility to compute composite event models during analysis. To best of our knowledge, no previous work in any known real-time analysis technique covers this aspect.

6.2 Task model: the request bound

The previous presented mathematical framework allows the formulation of advanced analysis techniques. Next, we discuss how to integrate the generalized multi-frame model and how easily interfering request bounds can be constructed to describe different scheduling policies.

6.2.1 Generalized request bound

The new approach to describe events with Dirac deltas is compelling: The advantage compared to established techniques is that the Dirac comb of Definition 12 addresses each event separately, and therefore, each event may have different properties. As a result, the model allows assigning different execution times to different events without any additional effort. In the established analysis, the request bound is given by a multiplication of the event bound and the worst-case execution time of the task. However, because it is easy to address each event separately by the unified event bound, the multiframe- [34] and the generalized multiframe model [8] integrates easily into the new approach. Formulating the event- and the request bound unified allows addressing each job with separate execution time. Therefore, it is possible to model task sets with complex execution time behaviour. However, often it is not necessary to assign an own execution time to each job. In this case, the execution time vector contains fewer elements as events occur by a task. Then the execution time can be addressed by restricted access to the given vector: The length of the vector then bounds the access as it could be described by $n \mod |C_{\tau,\epsilon}^n|$ as also given in the multiframe model:

Definition 17 (*Execution time vector*) The execution time vector introduced by [34] of k different execution times of a task is given by

$$C_{\tau,\epsilon}[n] = C_{\tau,\epsilon}^n = [c_1^+, \ldots, c_k^+] \tag{103}$$

Note the style of the notation: The idea is to address each component of the vector by n. If we like to address each event separately, it is not possible anymore to use the notation given in related work by defining request and demand bound functions. Addressing different events and jobs in one equation require to write an integral and two sum symbols every time. Therefore, it is necessary to introduce a short-form notation to simplify the writing and reading of event- and request bounds. Based on Einstein's well-known shorthand notation [20],[16] it is possible to define a shorthand notation for an event- and request bound that allows us to address each event or job of a given task separately:

Definition 18 (*Short form notation for request bounds*) Assuming $\delta(t' - \phi_\epsilon - np_\epsilon)_{\tau,\epsilon}^{n \leq k} \cdot C_{\tau,\epsilon}[n]$ is a short-hand notation for $\sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t - \phi_\epsilon - np_\epsilon) C_{\tau,\epsilon}[n]$ and C is a vector

[16] A detailed description gives the Appendix A.

that contains different execution times for different jobs, it is possible to write

$$\begin{aligned}
&\int_0^t \sum_{\epsilon\in\mathcal{E}_\tau}\sum_{n=0}^{k-1}\delta(t'-\phi_\epsilon-np_\epsilon)\cdot C_{\tau,\epsilon}[n]\cdot\overline{\mathbb{H}}\left(t'-t_a\right)\cdot\underline{\mathbb{H}}\left(t_b-t'\right)\,\mathrm{d}t' \\
&=\int_0^t \delta(t'-\phi_\epsilon-np_\epsilon)^{n\le k}_{\tau,\epsilon}\cdot C_{\tau,\epsilon}[n]\cdot\overline{\mathbb{H}}\left(t'-t_a\right)\cdot\underline{\mathbb{H}}\left(t_b-t'\right)\,\mathrm{d}t' \\
&=\int_{t_a}^{[t,t_b)} \delta(t'-\phi_\epsilon-np_\epsilon)^{n\le k}_{\tau,\epsilon}\cdot C_{\tau,\epsilon}[n]\,\mathrm{d}t' \\
&=\mathbb{E}^{n\le k}_{\tau,\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau,\epsilon}=\mathbb{R}^{n\le k}_{\tau,\epsilon}(t,\Delta^b_a)
\end{aligned}$$

To complete the integration of the multiframe model first introduced by [8] in this work, we need to redefine the concept of deadlines:

Definition 19 (*Deadline Vector*) The deadline vector is given by

$$\mathcal{D}_{\tau,\epsilon}[n]=\mathcal{D}^n_{\tau,\epsilon}=[d_1,\ldots,d_k] \tag{104}$$

6.2.2 The request bound of interfering jobs

The request bound function, as defined in general, does not distinguish between task priorities. Therefore it sums the requested execution times of all tasks. It is necessary to compute the interference of jobs to differentiate between the request of higher prior jobs that interrupt and interfere with a given job and other jobs that will have no impact on the final response. The following section will consider static as dynamic priorities as well. We look at how the same approach can solve both problems. Additionally, we find a unified solution of hierarchical scheduling of both algorithms which can be used in general to describe one of the two algorithms as well as a combination of them. Finally, the functions given in this section are solving the problems given by [43] and [12]. First, we formulate an abstract interfering request bound which can easily be adapted to different scheduling criteria:

Theorem 4 (Interference request bound) *Assume any criteria* $\square$ *and* $\blacksquare\geq\square$ *has a higher or equal priority and any job of* $\tau'_\blacksquare$ *interfere with* $\tau_\square$. *The interfering jobs execution time is selected by masking the request bound:*

$$\mathbb{R}^{\blacksquare\geq\square}_{\tau,\epsilon}(t,\Delta^b_a)=\mathbb{E}^{n\le k}_{\tau',\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau',\epsilon}\cdot\overline{\mathbb{H}}\left(\blacksquare_{\tau'}-\square_\tau\right) \tag{105}$$

Proof The Heaviside function as given by Definition 13 returns 1 if $\blacksquare_{\tau'}-\square_\tau\geq 0$ therefore

$$\int_{-\infty}^{\infty}\sum_{\epsilon\in\mathcal{E}_\tau}\sum_{n=0}^{k-1}\delta(t-\phi_\epsilon-np_\epsilon)\cdot C_{\tau,\epsilon}[n]\cdot\overline{\mathbb{H}}\left(\blacksquare_{\tau'}-\square_\tau\right)\,\mathrm{d}t \tag{106}$$

If a priority criterium of task τ' is higher than the criterium of task τ', then the task τ' interrupts τ, the Heaviside function becomes 1, and the request of the higher priority task is added to the request bound. If the criterium of task τ' is smaller than the one of task τ the Heaviside function is equal to 0 modelling no interrupt. □

The idea is to describe interference of jobs is generalized to a bunch of different relations $\mathbb{H}:\mathbb{R}\to\{0,1\}$ mapping any difference or real or integer numbers to boolean values:

Definition 20 (*Heaviside relation*) Again, assume any criteria $\square$, the main relations can be computed by the following heaviside functions:

$$\begin{aligned}
\blacksquare=\square &:= \overline{\mathbb{H}}\left(\blacksquare-\square\right)\cdot\overline{\mathbb{H}}\left(\square-\blacksquare\right)=\delta_{\blacksquare,\square}\\
\blacksquare\leq\square &:= \overline{\mathbb{H}}\left(\square-\blacksquare\right)\\
\blacksquare\geq\square &:= \overline{\mathbb{H}}\left(\blacksquare-\square\right)\\
\blacksquare<\square &:= \underline{\mathbb{H}}\left(\square-\blacksquare\right)\\
\blacksquare>\square &:= \underline{\mathbb{H}}\left(\blacksquare-\square\right)
\end{aligned}$$

The Kronecker delta $\delta_{\blacksquare,\square}$ is a well known short-form writing if criteria are equal.

Definition 21 (*Task scheduler*) Any boolean equation of Heaviside relations models a scheduler in real-time analysis because it defines whether two tasks interfere or not. Assume any Heaviside relation $\phi\in\{\mathbb{H}_{\blacksquare=\square},\mathbb{H}_{\blacksquare\leq\square},\mathbb{H}_{\blacksquare\geq\square},\mathbb{H}_{\blacksquare<\square},\mathbb{H}_{\blacksquare>\square}\}$ the function $\mathbb{S}:\Gamma^2\to\{0,1\}$ represents a task scheduler which describes the interference of two tasks:

$$\mathbb{S}^{\square}_{\tau,\tau'}:=\max\{\min_{\phi}\{x_\phi\}\} \tag{107}$$

Note, the operation max and min represents *or* and *and* on integers.

Example 5 (Static task scheduler) Assume static priorities as given in Definition 6. Two jobs interfere if

$$\mathbb{S}^{\pi}_{\tau,\tau'}:=\max\{\mathbb{H}_{\pi_\tau<\pi_{\tau'}},\min\{\mathbb{H}_{\pi_{\tau'}=\pi_\tau},\mathbb{H}_{t^r_{\tau'}<t^r_\tau}\}\} \tag{108}$$

In deadline monotonic scheduling the priority is not needed, it is possible to write directly

$$\mathbb{S}^{d}_{\tau,\tau'}:=\max\{\mathbb{H}_{d_{\tau'}<d_\tau},\min\{\mathbb{H}_{d_{\tau'}=d_\tau},\mathbb{H}_{t^r_{\tau'}<t^r_\tau}\}\} \tag{109}$$

$\mathbb{H}_{d_{\tau'}<d_\tau}$ and $\mathbb{H}_{d_{\tau'}=d_\tau}$ are disjunct, therefore

$$\mathbb{S}^{d}_{\tau,\tau'}:=\mathbb{H}_{\pi_{\tau'}<\pi_\tau}+\mathbb{H}_{\pi_{\tau'}=\pi_\tau}\cdot\mathbb{H}_{t^r_{\tau'}<t^r_\tau} \tag{110}$$

$$\mathbb{S}^{d}_{\tau,\tau'}:=\mathbb{H}_{d_{\tau'}<d_\tau}+\mathbb{H}_{d_{\tau'}=d_\tau}\cdot\mathbb{H}_{t^r_{\tau'}<t^r_\tau} \tag{111}$$

Suppose the upcoming analysis using interfering request bounds should support arbitrary deadlines, then we have to consider jobs with the same static priorities. Therefore jobs with the same priority interfere if the requested job starts later than the interfering job.[17]

Example 6 (Dynamic task scheduler) In dynamic scheduling the job with the earliest absolute deadline is scheduled. Therefore we have only to change the relative deadline to the absolute deadline in definition 5. In this case, the consideration of the request time is mandatory because system designers and programmers can not guarantee different absolute deadlines if the specified relative deadlines are different.

$$\mathbb{S}^{D^n}_{\tau,\tau'} := \max\{\mathbb{H}_{D^n_{\tau'}<D^n_\tau}, \min\{\mathbb{H}_{D^n_{\tau'}=D^n_\tau}, \mathbb{H}_{t^r_{\tau'}<t^r_\tau}\}\} \tag{112}$$

Note, that indifference to the static task scheduler the absolute deadline of each job must considered.[18] $\mathbb{H}_{D^n_{\tau'}<D^n_\tau}$ and $\mathbb{H}_{D^n_{\tau'}=D^n_\tau}$ are disjunct, therefore

$$\mathbb{S}^{D^n}_{\tau,\tau'} := \mathbb{H}_{D^n_{\tau'}<D^n_\tau} + \mathbb{H}_{D^n_{\tau'}=D^n_\tau} \cdot \mathbb{H}_{t^r_{\tau'}<t^r_\tau} \tag{113}$$

The first step in the discussion is the formulation of the interfering request bound for static schedulers and task priorities specified by fixed numbers:

Corollary 1 (Interference request bound in static scheduling) *Assume any static scheduler with a priority π_τ assigned to each task. If task τ' has a higher priority than task τ and a higher number of $\pi_{\tau'} > \pi_\tau$ specifies this behaviour, then the interference request bound $\mathbb{R}^{\pi_{\tau'}\geq\pi_\tau} : \Gamma^2 \times \mathbb{R}^2 \to \mathbb{R}$ is given by*

$$\mathbb{R}^{\pi_{\tau'}\geq\pi_\tau}_{\tau,\tau'}(t, \Delta^b_a) = \mathbb{E}^{n\leq k}_{\tau',\epsilon}(t, \Delta^b_a) \cdot C^n_{\tau',\epsilon} \cdot [\,\underline{\mathbb{H}}\,(\pi_{\tau'} - \pi_\tau) + \delta_{\pi_{\tau'},\pi_\tau} \cdot \overline{\mathbb{H}}\left(t^r_\tau - t^r_{\tau'}\right)\,] \tag{114}$$

Contrarily, if task τ' has a higher priority than task τ and a lower number of $\pi_{\tau'} < \pi_{\tau'}$ specifies this behaviour, then the priority difference in the equation changes. If we assume deadline monotone scheduling the interfering request bound can express this directly:

$$\mathbb{R}^{d_{\tau'}\leq d_\tau}_{\tau,\tau'}(t, \Delta^b_a) = \mathbb{E}^{n\leq k}_{\tau',\epsilon}(t, \Delta^b_a) \cdot C^n_{\tau',\epsilon} \cdot [\,\underline{\mathbb{H}}\,(d_\tau - d_{\tau'}) + \delta_{d_\tau,d_{\tau'}} \cdot \overline{\mathbb{H}}\left(t^r_\tau - t^r_{\tau'}\right)\,] \tag{115}$$

Proof Consider Theorem 4: For $\pi_{\tau'} > \pi_\tau$ the Heaviside function $\underline{\mathbb{H}}\,(\pi_{\tau'} - \pi_\tau) = 1$, and the execution request of task τ' is added to the interference task set of τ. If two priorities are equal the job with the earliest request is scheduled. The interference mask become one if $t^r_\tau - t^r_{\tau'} \geq 0$. The proof of the other DMS equation is obvious. □

According to this well-known definition of absolute deadlines, the interfering request bound in dynamic scheduling can be formulated by:

Corollary 2 (Interference request bound in dynamic scheduling) *The interfering request bound $\mathbb{R}^{D^n_{\tau'}\geq D^n_\tau} : \Gamma^2 \times \mathbb{R}^2 \to \mathbb{R}$ of higher priority tasks in dynamic scheduling is*

$$\mathbb{R}^{D^n_{\tau'}\leq D^n_\tau}_{\tau,\tau'}(t, \Delta^b_a) = \mathbb{E}^{n\leq k}_{\tau',\epsilon}(t, \Delta^b_a) \cdot C^n_{\tau',\epsilon} \cdot [\,\underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right) + \delta_{D^n_{\tau'},D^n_\tau} \cdot \overline{\mathbb{H}}\left(t^r_\tau - t^r_{\tau'}\right)\,] \tag{116}$$

Proof Assume dynamic scheduling and a given job $\tau_{\epsilon,n}$. The request bound of this job is the sum of all execution times of job's $\tau'_{\epsilon,n}$ with an absolute deadline shorter than the job's $\tau_{\epsilon,n}$ deadline. According to Theorem 4, the subtraction $D^n_\tau - D^n_{\tau'}$ is positive if $D^n_{\tau'} \leq D^n_\tau$, therefore in this case $\underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right) = 1$. If $D^n_{\tau'} > D^n_\tau$ the inequality $\underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right) = 0$. This means $\underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right)$ selects the higher priority jobs in dynamic scheduling. This approach models scheduling were any task instance with an absolute deadline smaller than the absolute deadline of the considered task instance interfere in the considered task. However, what happens if two instances have the same absolute deadline? In this case, a tie-breaking condition is needed. The instance with the smaller request is scheduled to avoid scheduling overhead.[19] This behaviour is modelled by $\delta_{D^n_{\tau'},D^n_\tau} \cdot \overline{\mathbb{H}}\left(t^r_\tau - t^r_{\tau'}\right)$. □

In real-time scheduling theory, the structure of equations changes on any new problem. As we demonstrated the unified theory, the request bound of interfering jobs can be expressed by just one equation choosing the Heaviside mask's correct parameters. It does not matter if we like to consider static or dynamic scheduling. Having just one theoretical approach is a decisive advantage compared to related work: While the structure of an equation does not change depending on the scheduling, it is possible to construct combined

[17] Such an assumption looks oversized. However, if we distinguish the jobs with different request times, arbitrary deadlines are integrated into the model for free. Note that it is possible to describe the established models by using the upper Heaviside function without a distinction of request times.

[18] In scheduling theory, we assume that any job could be executed if absolute deadlines are equal. It can be described again by the upper Heaviside function without any assumption about request times. Again, arbitrary deadlines could be modelled easily, considering request times.

[19] This assumption does not hold in general. However, if any job is scheduled if deadlines are equal only the Heaviside mask should be modified to model such a scheduling behavior. But this leads to an analysis over approximation. We will discuss this in detail later in section 6.7.1. example 7 and theorem 12.

schedulers and formulate their analysis by constructing new equations. As an example, let us consider a hierarchical static and dynamic or a mixed scheduler.

Theorem 5 (Interference request bound in hierarchical scheduling) *Assume a task set and assign a static priority to each task. Then tasks with different priorities schedule by static scheduling and tasks with equal priority schedule according to their deadlines dynamically. In the case of such hierarchical scheduling, the interfering request bound* $\mathbb{R}^{\tau' \geq \tau} : \Gamma^2 \times \mathbb{R}^2 \to \mathbb{R}$ *is*

$$\mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, \Delta^b_a) = \mathbb{E}^{n \leq k}_{\tau,\epsilon}(t, \Delta^b_a) \cdot C^n_{\tau,\epsilon} \cdot \max\left(\underline{\mathbb{H}}\left(\pi_{\tau'} - \pi_\tau\right), \delta_{\tau,\tau'} \cdot \left[\underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right) + \delta_{D^n_{\tau'}, D^n_\tau} \cdot \overline{\mathbb{H}}\left(t^r_{\tau'} - t^r_\tau\right) \right] \right) \tag{117}$$

Proof If a task has a higher priority than the considered task, the function $\underline{\mathbb{H}}(\pi_{\tau'} - \pi_\tau) = 1$ else it is 0. If the priority of the tasks is equal and the absolute deadline of the interfering task is smaller than the absolute deadline of the considered task the scheduling is described by $\delta_{\tau,\tau'} \cdot [\, \underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right) + \delta_{D^n_{\tau'}, D^n_\tau} \cdot \underline{\mathbb{H}}\left(t^r_{\tau'} - t^r_\tau\right) \,]$, as we already know from corollary 2. This function is only 1 if the absolute deadline of a potential interfering task τ' is shorter than the deadline of the considered task τ and the request time of the interfering task τ' is earlier than the request time of the considered task τ. Therefore, the selecting criteria to identify an interfering task leads to 0 or 1 dependently on the tasks priority or absolute deadline. The function $\max(\underline{\mathbb{H}}(\pi_{\tau'} - \pi_\tau)), \overline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right)) = 1$ if $\max(0, 1)$, $\max(0, 1)$ or $\max(1, 1)$. This implements an *or*-operation between static and dynamic scheduling. □

6.3 Analysis preliminaries

We need some additional assumptions to derive bound tests tests or a response time analysis based on the interfering request bound. In this section, we will introduce the concept of the remaining load to compute the backlog, which is not proceeded by a processor during a given time interval. Besides, we will give some useful definitions related to a generalized analysis framework.

Theorem 6 (Remaining load) *The remaining load of a job interfered with other jobs is the computational demand of a given time interval* $[0, t)$ *which cannot be computed by the processor during this time interval. Assume that the timing interval* $\Delta^t_0 = t$*, then the remaining load* $\mathbb{L}^{\tau' \geq \tau} : \Gamma^2 \times \mathbb{R} \to \mathbb{R}$ *is:*

$$\mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) = \max_{0 \leq s \leq t} \{\mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, t) - \mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(s, t) - t\} \tag{118}$$

Proof If $\forall t \in \mathbb{R} : \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) \geq 0$ und $\mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) = \mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, t) - t$, then

$$\mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, t) - \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) - t \leq 0 \tag{119}$$

$$\mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t - s, t) - \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t - s) - t \leq 0 \tag{120}$$

$$\mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, t) - \mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(s, t) - \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) - \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(s) - t \leq 0 \tag{121}$$

Because $\mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) - \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(s) \leq \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t - s)$ ([13], p. 7) the following equation holds:

$$\mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) \geq \mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, t) - \mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(s, t) - t \tag{122}$$

$$\mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) = \sup_{0 \leq s \leq t} \{\mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, t) - \mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(s, t) - t\} \tag{123}$$

At this point, we can carefully review the properties of the unified event bound as given by Theorem 1: Note that the domain of this function is a compact space: Because we defined the limits of the integral as an open interval, the domain t is compact. If we define any analysis in a bounded domain, then the unified event bound and therefore, the request and demand bound are compact. Bounding periodic task sets to their hyper-period bounds the timing interval as well. Therefore, the supremum of the function is equal to its maximum: $sup = max$. Then

$$\mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t) = \max_{0 \leq s \leq t} \{\mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(t, t) - \mathbb{R}^{\tau' \geq \tau}_{\tau,\tau'}(s, t) - t\} \tag{124}$$

□

The proof builds on the leaky bucket algorithm. In the case no load is requested to a processor, the remaining load is equal to 0. Only if service is requested, the processor executes it with the rate of t. The above proof is directly adapted from the network calculus as given by ([13], p.10, f.). By applying the leaky bucket approach from network calculus to the problem of remaining load in real-time analysis is elegant and leads to a beautiful result. Because we defined a compact unified event-, request- and demand bound by using an integral and we model its limits by a Heaviside function, we can now combine the result of the network respective the real-time calculus with the work done in established scheduling theory. If supremum and infimum become maximum and minimum in all cases, we can further use effective maximization and minimization techniques supported by numerical mathematics and therefore it is easy to apply this theory to computer algebra systems or numerical math tools.

Definition 22 (*Average load*) Given any time interval $[a, b]$[20], the average load $\Delta_0^t = t$, the remaining load $U : \Gamma \times \mathbb{R} \to \mathbb{R}$ is the mean value of the requested load related to the interval. The average load of a task set is the sum of the average loads of each task.

$$\begin{aligned}\mathbb{U}_\Gamma(\Delta_a^b) &= \sum_{\tau\in\Gamma}\mathbb{U}_\tau(\Delta_a^b)\\ &= \frac{1}{b-a}\cdot\left(\mathbb{L}_\Gamma(a) + \int_a^b \sum_{\tau\in\Gamma}\left(\mathrm{m}_\tau^k\cdot c_\tau^+\right)\,\mathrm{d}t'\right)\end{aligned} \tag{125}$$

Lemma 6 (Average load by the unified request bound) *The average load in any time interval* $[a, b)$ *of a task set on a processor is given by*

$$\mathbb{U}_\Gamma(\Delta_a^b) = \frac{\mathbb{L}_\Gamma(a) + \mathbb{R}_\Gamma(\Delta_a^b, \Delta_a^b)}{b-a} \tag{126}$$

and in the special case in the interval $[0, t)$

$$\mathbb{U}_\Gamma(\Delta_0^t) = \mathbb{U}_\Gamma(t) = \frac{\mathbb{R}_\Gamma(t,t)}{t} \tag{127}$$

Now only tasks with the same or a higher priority should be considered, we use

$$\mathbb{U}_{\tau,\tau'}^{\tau'\geq\tau}(\Delta_a^b) = \frac{\mathbb{L}_{\tau,\tau'}^{\tau'\geq\tau}(a) + \mathbb{R}_{\tau,\tau'}^{\tau'\geq\tau}(\Delta_a^b, \Delta_a^b)}{b-a} \tag{128}$$

and

$$\mathbb{U}_{\tau,\tau'}^{\tau'\geq\tau}(t) = \frac{\mathbb{R}_{\tau,\tau'}^{\tau'\geq\tau}(t,t)}{t} \tag{129}$$

for the interval $[0, t)$.

Proof The average load of the requested jobs in any interval related to the duration of this interval. In some cases, there is some load left from previous intervals. That remains in an additional load:

$$\mathbb{U}_\Gamma(\Delta_a^b) = \frac{1}{\Delta_a^b}\cdot\left(\mathbb{L}_\Gamma(a) + \int_{\Delta_a^b}\sum_{\tau\in\Gamma}\mathrm{m}_\tau^k\cdot c_\tau^+\,\mathrm{d}t\right) \tag{130}$$

$$= \frac{1}{b-a}\cdot\left(\mathbb{L}_\Gamma(a) + \int_a^b \sum_{\tau\in\Gamma}\mathrm{m}_\tau^k\cdot c_\tau^+\,\mathrm{d}t\right) \tag{131}$$

$$= \frac{\mathbb{L}_\Gamma(a) + \mathbb{R}_\Gamma(\Delta_a^b, \Delta_a^b)}{b-a} \tag{132}$$

Note that the computation of the utilization requires a summation during $[a, b)$. Therefore, the request bound is given by $\mathbb{R}_\Gamma(\Delta_a^b, \Delta_a^b)$. Let us now consider the utilization in the interval $[0, t)$:

$$\mathbb{U}_\Gamma(t) = \sum_{\tau\in\Gamma}\mathbb{U}_\tau(t) \tag{133}$$

$$= \sum_{\tau\in\Gamma}\left(\frac{1}{t}\cdot\int_0^t \mathrm{m}_\tau^k\cdot c_\tau^+\,\mathrm{d}t'\right) \tag{134}$$

$$= \frac{1}{t}\cdot\int_0^t\sum_{\tau\in\Gamma}\left(\mathrm{m}_\tau^k\cdot c_\tau^+\right)\,\mathrm{d}t' \tag{135}$$

$$= \frac{\mathbb{R}_\Gamma(t,t)}{t} \tag{136}$$

□

6.4 Unified bound tests analysis

bound tests tests build on utilization bounds. Therefore we have to check whether the utilization of task set is always smaller than 1 or 100%. A utilization bound given for any time interval based on the interference request bound, and the average load allows bound tests tests for static, dynamic, and hierarchical scheduling.

Theorem 7 (bound tests analysis for static and dynamic scheduling) *Assume a task set and a hierarchical scheduler. If a task has a higher priority than another task, it executes first, and if two tasks have the same priority, they are scheduling under earliest deadline first. The bound tests of a given independent task set executed by one computing resource with a hierarchical static, and a dynamic scheduler can then be guaranteed, if and only if*

$$\forall t \in \mathcal{P}_\Gamma : \quad \mathbb{R}_{\tau,\tau'}^{\pi_{\tau'}\geq\pi_\tau}(t, \mathcal{P}_\Gamma) + \mathbb{R}_{\tau,\tau'}^{\pi_{\tau'}\geq\pi_\tau}(t, \mathcal{P}_\Gamma) \leq t \tag{137}$$

Proof The bound tests test can be derived from the utilization bound[21]

$$U_\Gamma(\Delta_a^b) = \frac{\mathbb{L}_\Gamma(t) + \mathbb{R}_\Gamma(\Delta_a^b, \Delta_a^b)}{t_b - t_a} \tag{138}$$

and in the special case in the interval $[0, t)$

$$U_\Gamma(\Delta_0^t) = U_\Gamma(t) = \frac{\mathbb{R}_\Gamma(t,t)}{t} \tag{139}$$

[20] We assume only to count events in $[0, t)$ as given by our unified request bound. However, to define a utilization interval a closed interval is needed.

[21] To keep the proof simple we do not consider the request times.

A task set is feasible if $\forall t \in [0, \mathcal{P}_\Gamma]$ for any task the utilization $u_\tau(t) \leq 1$. If $t_a = t_0 = 0$ then $\mathbb{L}(0) = 0$ and therefore[22]

$$\forall t \in \mathcal{P}_\Gamma : \quad \frac{\mathbb{R}_{\tau,\Gamma}^{\pi_{\tau'} \geq \pi_\tau}(t, \mathcal{P}_\Gamma)}{t} \leq 1 \tag{140}$$

Now we separate all higher priority tasks from tasks with the same priority and multiply by t:

$$\forall t \in \mathcal{P}_\Gamma : \quad \mathbb{R}_{\tau,\Gamma}^{\pi_{\tau'} = \pi_\tau}(t, \mathcal{P}_\Gamma) + \mathbb{R}_{\tau,\Gamma}^{\pi_{\tau'} > \pi_\tau}(t, t) \leq t \tag{141}$$

or

$$\forall t \in \mathcal{P}_\Gamma, \forall \tau \in \Gamma : \quad \sum_{\tau' \in \Gamma} \mathbb{R}_{\tau'}(t - d_{\tau'}, \mathcal{P}_\Gamma) \cdot \delta_{\pi_{\tau'}, \pi_\tau} \tag{142}$$

$$+ \sum_{\tau' \in \Gamma} \mathbb{R}_{\tau'}(t, t) \cdot \underline{\mathbb{H}}(\pi_{\tau'} - \pi_\tau) \leq t \tag{143}$$

Note that we consider tasks with the same priority and tasks with higher priorities in independent terms because we want to derive a bound tests test for static and dynamic scheduling as well. Therefore we have to consider two cases:

A If $\pi_{\tau'} = \pi_\tau$ then $\delta_{\pi_{\tau'}, \pi_\tau} = 1$ and $\underline{\mathbb{H}}(\pi_{\tau'} - \pi_\tau) = 0$. The bound tests test for dynamic scheduling is then given by

$$\forall t \in \mathcal{P}_\Gamma, \forall \tau \in \Gamma : \quad \sum_{\tau' \in \Gamma} \mathbb{E}_{\tau'}(t - d_{\tau'}, \mathcal{P}_\Gamma) \cdot c_{\tau'}^+$$

$$= \sum_{\tau' \in \Gamma} \mathbb{R}_{\tau'}(t - d_{\tau'}, \mathcal{P}_\Gamma) \leq t \tag{144}$$

In this case $\tau' = \tau$ and therefore

$$\forall t \in \mathcal{P}_\Gamma : \sum_{\tau \in \Gamma} \mathbb{E}_\tau(t - d_\tau, \mathcal{P}_\Gamma) \cdot c_{\tau'}^+$$

$$= \sum_{\tau \in \Gamma} \mathbb{R}_\tau(t - d_\tau, \mathcal{P}_\Gamma) \leq t \tag{145}$$

which is equal to the processor demand test or Problem 2.

B Consider $\pi_{\tau'} \neq \pi_\tau$: Now $\delta_{\pi_{\tau'}, \pi_\tau} = 0$ only for the considered task τ and in all other cases $\delta_{\pi_{\tau'}, \pi_\tau} = 0$ and $\underline{\mathbb{H}}(\pi_{\tau'} - \pi_\tau) = 1$ if a task τ' has a higher priority than the considered task τ. Then we get

$$\forall t \in \mathcal{P}_\Gamma, \forall \tau \in \Gamma : \quad \mathbb{E}_\tau(t - d_\tau, \mathcal{P}_\Gamma) \cdot c_\tau^+$$

$$+ \sum_{\tau' \in \overline{\Gamma}_\tau} \mathbb{E}_{\tau'}(t, t) \cdot c_\tau^+ \leq t \tag{146}$$

or

$$\forall t \in \mathcal{P}_\Gamma, \forall \tau \in \Gamma : \quad \mathbb{R}_\tau(t - d_\tau, \mathcal{P}_\Gamma)$$

$$+ \sum_{\tau' \in \overline{\Gamma}_\tau} \mathbb{R}_{\tau'}(t, t) \leq t \tag{147}$$

where τ is the considered task and τ' are all higher priority tasks. This result is equivalent to

$$\forall t \in \mathcal{P}_\Gamma, \forall \tau \in \Gamma : \quad \mathbb{R}_\tau(t - d_\tau, \mathcal{P}_\Gamma) \leq t - \sum_{\tau' \in \overline{\Gamma}_\tau} \mathbb{R}_{\tau'}(t, t) \tag{148}$$

which is equal to the processor demand test for static scheduling originally given by [6]. □

6.5 Unified response time analysis

In this section, we will derive a response time analysis based on the average load and the unified event bound to simplify the mathematical framework as given by related work. As a result of the section, we will see that the unified event bound solves Problem 3.

Theorem 8 (Unified response time analysis) *If a job is scheduled by any scheduling algorithm assuming priorities given by any relation between two different variables* $\blacksquare \geq \square$, *the request time of the job is* $t_{\tau,\epsilon}$. *Let the response time* $r_{\tau,\epsilon} = t_{\tau,\epsilon}^f - t_{\tau,\epsilon}^r$ *be the difference of the finishing time* $t_{\tau,\epsilon}^f$ *and the request time. The response time* $r_{\tau,\epsilon}$ *is bounded by*

$$\forall \tau_\epsilon \in \Gamma : \quad \mathbb{L}_{\tau,\tau'}^{\blacksquare \geq \square}(t_{\tau,\epsilon}^r) + \mathbb{R}_{\tau,\tau'}^{\blacksquare \geq \square}(r_{\tau,\epsilon}, r_{\tau,\epsilon}) - r_{\tau,\epsilon} = 0 \tag{149}$$

Proof Again, we start with the average load. The average load of any given time interval $[t_a, t_b)$ is given by:

$$\mathbb{U}_\Gamma(\Delta_a^b) = \frac{\mathbb{L}_\Gamma(a) + \mathbb{R}_\Gamma(\Delta_a^b, \Delta_a^b)}{\Delta_a^b} \leq 1 \tag{150}$$

Now, we consider the interval $\Delta_a^b = t_{\tau,\epsilon}^f - t_{\tau,\epsilon}^r = r_{\tau,\epsilon}$ for each job of all tasks, therefore

$$\forall \tau_\epsilon \in \Gamma : \quad \frac{\mathbb{L}_\Gamma(t_{\tau,\epsilon}^r) + \mathbb{R}_\Gamma(r_{\tau,\epsilon}, r_{\tau,\epsilon})}{r_{\tau,\epsilon}} \leq 1 \tag{151}$$

We only have to consider all tasks of the same or a higher priority than the considered task's priority:

$$\forall \tau_\epsilon \in \Gamma : \frac{\mathbb{L}_{\tau,\tau'}^{\blacksquare \geq \square}(t_{\tau,\epsilon}^r) + \mathbb{R}_{\tau,\tau'}^{\blacksquare \geq \square}(r_{\tau,\epsilon}, r_{\tau,\epsilon})}{r_{\tau,\epsilon}} \leq 1 \tag{152}$$

During a busy period, the average load is positive and more significant than 100% because the requested execution

[22] Remember $\mathcal{P}_\Gamma$ is the hyperperiod of the considered task set.

demand is higher than the elapsed processor time. The average load will be smaller than 100% if the requested demand in a time interval is smaller than the processing time interval. In this case, the processor is idle. Therefore, the end of the busy period is exact if the average load is equal to 100%:

$$\forall \tau_\epsilon \in \Gamma: \quad \frac{\mathbb{L}^{\blacksquare \geq \square}_{\tau,\tau'}(t^r_{\tau,\epsilon}) + \mathbb{R}^{\blacksquare \geq \square}_{\tau,\tau'}(r_{\tau,\epsilon}, r_{\tau,\epsilon})}{r_{\tau,\epsilon}} = 1 \tag{153}$$

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}^{\blacksquare \geq \square}_{\tau,\tau'}(t^r_{\tau,\epsilon}) + \mathbb{R}^{\blacksquare \geq \square}_{\tau,\tau'}(r_{\tau,\epsilon}, r_{\tau,\epsilon}) = r_{\tau,\epsilon} \tag{154}$$

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}^{\blacksquare \geq \square}_{\tau,\tau'}(t^r_{\tau,\epsilon}) + \mathbb{R}^{\blacksquare \geq \square}_{\tau,\tau'}(r_{\tau,\epsilon}, r_{\tau,\epsilon}) - r_{\tau,\epsilon} = 0 \tag{155}$$

As we want to compute the response time $r_{\tau,\epsilon}$, we also except task sets with an average load equal to 100%, which means after a job has finished the next higher priority job starts immediately, and the processor does not idle. As a result, we have to end the summation of task requests exactly at $r_{\tau,\epsilon}$ and we get

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}^{\blacksquare \geq \square}_{\tau,\tau'}(t^r_{\tau,\epsilon}) + \mathbb{R}^{\blacksquare \geq \square}_{\tau,\tau'}(r_{\tau,\epsilon}, r_{\tau,\epsilon}) - t^r_{\tau,\epsilon} = 0 \tag{156}$$

Note that in the worst-case in static scheduling we only have to consider the first job of each task. The remaining load then is $\mathbb{L}^{\pi' \geq \pi}_{\tau,\tau'}(0) = 0$ and the worst-case response time becomes

$$\forall \tau \in \Gamma: \quad : \mathbb{R}^{\blacksquare \geq \square}_{\tau,\tau'}(r^+_{\tau,\epsilon}, r^+_{\tau,\epsilon}) - r^+_{\tau,\epsilon} = 0 \tag{157}$$

□

Theorem 8 describes an unified abstract form of the busy window approach. In real-time scheduling theory, static and dynamic scheduling are major scheduling algorithms. Therefore, the unified approach has to be adapted to static as well as to dynamic scheduling [23]:

Corollary 3 (Static response time analysis) *Assume a given task set with static priorities. The abstract given relation* $\blacksquare \geq \square$ *is then replaced by* $\pi_{\tau'} \geq \pi_\tau$ *formulating the static priority scheme. The response time then becomes*

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}^{\pi_{\tau'} \geq \pi_\tau}_{\tau,\tau'}(t_{\tau,\epsilon}) + \mathbb{R}^{\pi_{\tau'} \geq \pi_\tau}_{\tau,\tau'}(r_{\tau,\epsilon}, r_{\tau,\epsilon}) - r_{\tau,\epsilon} = 0 \tag{158}$$

Proof Replacing $\blacksquare \geq \square$ with $\pi_{\tau'} \geq \pi_\tau$, the proof follows directly from Theorems 1 and 8. □

Corollary 3 solves Problem 3.

Corollary 4 (Dynamic response time analysis) *Assume a given task set with dynamic priorities. The abstract given relation* $\blacksquare \geq \square$ *is then replaced by* $D^n_{\tau'} \leq D^n_\tau$*, formulating the dynamic priority scheme corresponding to EDF scheduling. The response time then becomes*

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}^{D^n_{\tau'} \leq D^n_\tau}_{\tau,\tau'}(t_{\tau,\epsilon}) + \mathbb{R}^{D^n_{\tau'} \leq D^n_\tau}_{\tau,\tau'}(r_{\tau,\epsilon}, r_{\tau,\epsilon}) - r_{\tau,\epsilon} = 0 \tag{159}$$

Proof Replacing $\blacksquare \geq \square$ with $D^n_{\tau'} \leq D^n_\tau$, the proof follows directly from Theorems 2 and 8. □

A logical combination of the two selecting Heaviside functions describes a hierarchical scheduler, as shown in Theorem 5.

Corollary 5 *(Hierarchical response time analysis) Assume a scheduler which scheduled tasks by their given priorities and all tasks with the same priority by their deadline. A hierarchical busy window response time analysis is given by*

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}^{\leq}_{\tau,\tau'}(t_{\tau,\epsilon}) + \mathbb{R}^{\leq}_{\tau,\tau'}(r_{\tau,\epsilon}, r_{\tau,\epsilon}) - r_{\tau,\epsilon} = 0 \tag{160}$$

Proof The analysis directly follows from Theorems 5 and 8. □

6.6 Relationship to related work

After discussing the unified approach to static and dynamic bound tests and response time analysis, we will consider its relationship to prior work. The question of this section is if the new methodology can express advanced techniques in the real-time analysis as extensions to the busy window approach introduced by [29]. In detail, we will discuss the model of bursty or sporadic events and arbitrary deadlines [48], the relative complex analysis of response time in dynamic scheduling as discussed by [37,41]. Note that in the original work the analysis equations are derivate geometrically from Gantt charts. In this work, we have formalized the Gantt chart approach algebraic. Therefore it should be possible to show the mathematical relationship of recent work to the new algebraic approach and how the new unified theory help in proving theorems. The algebraic method is compatible with related work and can be easily used in future work to proof new theorems or to adapt specula case results to the general model.

6.7 Bursty events in response time analysis

Tindell et al. [48] investigates the response time analysis of bursty event sequences and gives an event bound for this so-called sporadic event model:

[23] Including other scheduling schemes should be future work.

$$\mathbb{E}_\tau(t) = \left\lfloor \frac{t}{p_o} \right\rfloor b + \min\left(\left\lceil \frac{t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o}{p_i} \right\rceil, b \right) \tag{161}$$

Now, we can show how this equation is related to the new approach. First, we have to consider both parts of the equation separately to combine them later in the theorem:

Lemma 7 (Counting full bursts) *The first part of the equation for the event bound of sporadic tasks describes the number of fully covered bursts and multiplies the number of them by the maximal number of events per burst. The unified event bound describes it as*

$$\left\lfloor \frac{t}{p_o} \right\rfloor b = \int_0^{\left\lfloor \frac{t}{p_o} \right\rfloor p_o} \left(\sum_{n=0}^{\infty} \delta(t' - np_o) \cdot \sum_{m=0}^{b-1} \delta(t' - mp_I) \right) \mathrm{d}t' \tag{162}$$

Proof If we describe $\left\lfloor \frac{t}{p_o} \right\rfloor b$ by a Dirac comb we find

$$\left\lfloor \frac{t}{p_o} \right\rfloor b = \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \delta(t' - np_o) \cdot b \cdot \underline{\mathbb{H}}\left(t'\right) \cdot \overline{\mathbb{H}}\left(\left\lfloor \frac{t}{p_o} \right\rfloor p_o - t' \right) \mathrm{d}t' \tag{163}$$

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \delta(t' - np_o) \sum_{m=0}^{b-1} \delta(t' - mp_i) \overline{\mathbb{H}}\left(t'\right) \cdot \overline{\mathbb{H}}\left(\left\lfloor \frac{t}{p_o} \right\rfloor p_o - t' \right) \mathrm{d}t' \tag{164}$$

$$= \int_0^{\left\lfloor \frac{t}{p_o} \right\rfloor p_o} \sum_{n=0}^{\infty} \delta(t' - np_o) \cdot \sum_{m=0}^{b-1} \delta(t' - mp_i) \, \mathrm{d}t' \tag{165}$$

□

Lemma 8 (Counting part bursts) *In some cases, the last burst does not fit in the time interval examined. In this case, just events of a part of the burst are count:*

$$\min\left(\left\lceil \frac{t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o}{p_i} \right\rceil, b \right) = \int_{-\infty}^{\infty} \sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \underline{\mathbb{H}}\left(t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o \right) \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \tag{166}$$

Proof The second part of the event bound is given by $\min\left(\left\lceil \frac{t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o}{p_i} \right\rceil, b \right)$. We therefore have to enumerate all events between $\left\lfloor \frac{t_0}{p_0} \right\rfloor p_0$ and t. These events occour with the inner period p_i and we have to count

$$\int_{-\infty}^{\infty} \sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \underline{\mathbb{H}}\left(t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o \right) \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \tag{167}$$

events. □

After showing how the two parts of the sporadic model fit to the unified event bound, the final theorem can be formulated:

Theorem 9 (Unified response time analysis for the bursty or sporadic event model) *The sporadic or event model can be derived from event streams:*

$$\left\lfloor \frac{t}{p_o} \right\rfloor b + \min\left(\left\lceil \frac{t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o}{p_i} \right\rceil, b \right) = \int_{-\infty}^{\infty} \hat{u}^{k,l}_{\epsilon_o,\epsilon_i} \cdot \overline{\mathbb{H}}\left(t'\right) \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \tag{168}$$

Proof Applying Lemmas 7 and 8, the number of events counted by the event bound of the bursty or sporadic event model writes

$$\mathbb{E}_\tau(t) = \left\lfloor \frac{t}{p_o} \right\rfloor b + \min\left(\left\lceil \frac{t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o}{p_i} \right\rceil, b \right) \tag{169}$$

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \delta(t' - np_o) \sum_{m=0}^{b-1} \delta(t' - mp_i) \overline{\mathbb{H}}\left(t'\right) \cdot \overline{\mathbb{H}}\left(\left\lfloor \frac{t}{p_o} \right\rfloor p_o - t' \right) \mathrm{d}t' \tag{170}$$

$$+ \int_{-\infty}^{\infty} \sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \underline{\mathbb{H}}\left(t' - \left\lfloor \frac{t}{p_o} \right\rfloor p_o \right) \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \tag{171}$$

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \delta(t' - np_o) \cdot \sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \overline{\mathbb{H}}\left(t'\right) \cdot \overline{\mathbb{H}}\left(\left\lfloor \frac{t}{p_o} \right\rfloor p_o - t' \right) \tag{172}$$

$$+ \sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \underline{\mathbb{H}}\left(t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o \right) \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \tag{173}$$

While $\sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \overline{\mathbb{H}}(t') \cdot \overline{\mathbb{H}}\left(\left\lfloor \frac{t}{p_o} \right\rfloor p_o - t'\right)$ counts all inner events until $\left\lfloor \frac{t}{p_o} \right\rfloor p_o$ and $\sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \underline{\mathbb{H}}\left(t - \left\lfloor \frac{t}{p_o} \right\rfloor p_o\right) \cdot \underline{\mathbb{H}}(t - t')$ counts all inner events from $\left\lfloor \frac{t}{p_o} \right\rfloor p_o$ to t both parts can be substituted by $\sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \overline{\mathbb{H}}(t') \cdot \underline{\mathbb{H}}(t - t')$:

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \delta(t' - np_o) \cdot \sum_{m=0}^{b-1} \delta(t' - mp_i) \cdot \overline{\mathbb{H}}(t') \cdot \underline{\mathbb{H}}(t - t')\, \mathrm{d}t' \tag{174}$$

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \sum_{m=0}^{b-1} \delta(t' - np_o) \cdot \delta(t' - mp_i) \cdot \overline{\mathbb{H}}(t') \cdot \underline{\mathbb{H}}(t - t')\, \mathrm{d}t' \tag{175}$$

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \sum_{m=0}^{b-1} \delta(t' - np_o - mp_i) \cdot \overline{\mathbb{H}}(t') \cdot \underline{\mathbb{H}}(t - t')\, \mathrm{d}t' \tag{176}$$

Assume $\phi_o = 0$ and $\phi_i = 0$ and the inner period $p_{\epsilon_i} = p_i$ and the outer period is $p_{\epsilon_o} = p_o$. It can be seen that the sporadic event model is a special case of the event spectrum as computed by Theorem 3:

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \sum_{m=0}^{b-1} \delta(t' - \phi_o - \phi_i - np_o - mp_i) \cdot \overline{\mathbb{H}}(t') \cdot \underline{\mathbb{H}}(t - t')\, \mathrm{d}t' \tag{177}$$

$$= \int_{-\infty}^{\infty} \sum_{n=0}^{\infty} \sum_{m=0}^{b-1} \delta(t' - \phi_{\epsilon_1} - \phi_{\epsilon_2} - np_{\epsilon_o} - mp_{\epsilon_i}) \cdot \overline{\mathbb{H}}(t') \cdot \underline{\mathbb{H}}(t - t')\, \mathrm{d}t' \tag{178}$$

$$= \int_{-\infty}^{\infty} \hat{\mathbb{m}}_{\epsilon_o, \epsilon_i}^{k,l} \cdot \overline{\mathbb{H}}(t') \cdot \underline{\mathbb{H}}(t - t')\, \mathrm{d}t' \tag{179}$$

with $k = \infty$ and $l = b$ which proves the theorem. □

As second [48] covers in his work are the question of what happens if deadlines are longer than the interarrival time of jobs. Such an extension generalizes the analysis approach and opens real-time analysis to a wide range of industrial applications.

Theorem 10 (Unified response time analysis for arbitrary deadlines) *The unified response time analysis can be used to derive the analysis for arbitrary deadlines as given in* [48]*:*

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}_{\tau,\tau'}^{\pi' \geq \pi}(t_{\tau,\epsilon}^r) + \mathbb{R}_{\tau,\tau'}^{\pi' \geq \pi}(r_{\tau,\epsilon}, \Delta_{t_{\tau,\epsilon}^r}^{r_{\tau,\epsilon}}) - r_{\tau,\epsilon} = 0 \tag{180}$$

is identical to

$$w_\tau(q) = (q+1) \cdot c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \left\lceil \frac{w_\tau(q)}{p_{\tau'}} \right\rceil \cdot c_{\tau'}^+ \tag{181}$$

for all jobs in the first busy interval.

Proof Applying the general form to the response time analysis to find response times for tasks with arbitrary deadlines we assume that during a busy period more than one job of the considered task will start:

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}_{\tau,\tau'}^{\pi' \geq \pi}(t_{\tau,\epsilon}^r) + \mathbb{R}_{\tau,\tau'}^{\pi' \geq \pi}(r_{\tau,\epsilon}, \Delta_{t_{\tau,\epsilon}^r}^{r_{\tau,\epsilon}}) - r_{\tau,\epsilon} = 0 \tag{182}$$

Jobs in [48] are denoted with the symbol q instead of ϵ in this work:

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{L}_{\tau,\tau'}^{\pi' \geq \pi}(t_{\tau,q}) + \mathbb{R}_{\tau,\tau'}^{\pi' \geq \pi}(r_{\tau,q}, \Delta_{t_{\tau,q}^r}^{r_{\tau,q}}) - r_{\tau,q} = 0 \tag{183}$$

By definition, the average load in the considered interval $[0, t_{\tau,q}]$ is greater than 1, therefore $\mathbb{L}_{\tau,\tau'}^{\pi' \geq \pi}(t_{\tau,q}) = \mathbb{R}_{\tau,\tau'}^{\pi' \geq \pi}(r_{\tau,q}, \Delta_0^{t_{\tau,q}^r}) - t_{\tau,q}^r$ and we get

$$\forall \tau_\epsilon \in \Gamma: \quad \mathbb{R}_{\tau,\tau'}^{\pi' \geq \pi}(r_{\tau,q}, \Delta_0^{t_{\tau,q}^r}) - t_{\tau,q}^r + \mathbb{R}_{\tau,\tau'}^{\pi' \geq \pi}(r_{\tau,q}, \Delta_{t_{\tau,q}^r}^{r_{\tau,q}}) - r_{\tau,q} = 0 \tag{184}$$

Assuming periodic bursts and the periodic event model: If a fixed number of jobs with the same priority and a fixed period a_τ occur during the interval $\Delta_0^{t_{\tau,q}^r}$, the request bound of all jobs is:

$$\sum_{\tau' \in \Gamma} \mathbb{R}_{\tau'}(t_{\tau,q}^r, \Delta_0^{t_{\tau,q}^r}) \cdot \delta_{\pi_\tau, \pi_{\tau'}} = \left\lceil \frac{t_{\tau,q}^r}{a_\tau} \right\rceil \cdot c_\tau^+ \tag{185}$$

and according to Lemma 4, Eq. 184 is equal to

$$r_{\tau,q} = \left\lceil \frac{t_{\tau,q}^r}{a_\tau} \right\rceil \cdot c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \left\lceil \frac{t_{\tau,q}^r}{p_{\tau'}} \right\rceil \cdot c_{\tau'}^+ - t_{\tau,q}^r + c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \left\lceil \frac{r_{\tau,q}}{p_{\tau'}} \right\rceil \cdot c_{\tau'}^+ \tag{186}$$

$$r_{\tau,q} = \left\lceil \frac{t_{\tau,q}^r}{a_\tau} \right\rceil \cdot c_\tau^+ - t_{\tau,q}^r + c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \left\lceil \frac{t_{\tau,q} + r_{\tau,q}}{p_{\tau'}} \right\rceil \cdot c_{\tau'}^+ \tag{187}$$

$$r_{\tau,q} = \left\lceil \frac{qa_\tau}{a_\tau} \right\rceil \cdot c_\tau^+ - qa_\tau + c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \left\lceil \frac{qa_\tau + r_{\tau,q}}{p_{\tau'}} \right\rceil \cdot c_{\tau'}^+ \tag{188}$$

All response times during the busy interval are given by $r_\tau = \{\forall q \in \mathbb{N} : w_\tau(q) - q \cdot a_\tau\}$ and $w_\tau(q) = r_{\tau,q} + q \cdot a_\tau$. We substitute $r_{\tau,q} + q \cdot a_\tau$:

$$w_\tau(q) = qc_\tau^+ + c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \left\lceil \frac{w_\tau(q)}{p_{\tau'}} \right\rceil \cdot c_{\tau'}^+ \tag{189}$$

$$w_\tau(q) = (q+1) \cdot c_\tau^+ + \sum_{\tau' \in \overline{\Gamma}_\tau} \left\lceil \frac{w_\tau(q)}{p_{\tau'}} \right\rceil \cdot c_{\tau'}^+ \tag{190}$$

As $r_\tau^+ = \max_{\forall q \in \mathbb{N}} \{w_\tau(q) - q \cdot a_\tau\}$ we derived the test for arbitrary deadlines from the unified model. □

As a result, we found that the well-known sporadic or bursty event model and the analysis for arbitrary deadlines is a particular case of the unified event bound approach.

6.7.1 Response time analysis in dynamic scheduling

The sporadic and arbitrary model developed by [48] was applied to dynamic scheduling by [41]. The work of [37] generalizes [41] approach. The proof of the following theorem shows that the new theory can express the previous analysis by the same equations of the unified approach as used in the static scheduling analysis.

Theorem 11 (Equivalence of Spuri's dynamic analysis) *The response time analysis extension to the arbitrary busy window approach as given by Spuri is equivalent to the unified response time analysis:*

$$\begin{aligned} w_{\tau'}(t, D_\tau) &= \min \left\{ \left\lceil \frac{t}{p_{\tau'}} \right\rceil, \left\lfloor \frac{D_\tau - d_{\tau'}}{p_{\tau'}} \right\rfloor \right\} \\ &= \int_0^t \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \cdot \overline{\mathbb{H}}\left(D_\tau^n - D_{\tau'}^n\right) \\ &\quad \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \end{aligned} \tag{191}$$

Proof To simplify the equations, we only consider the interfering request bound of one task with all others. There is no loss of generality while the busy window is the sum of all tasks request bounds. Because of Lemmas 4 and 5 Spuri's busy window can be written as

$$\begin{aligned} w_{\tau'}(t, D_\tau) = \min \Bigg\{ & \int_0^t \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t', \\ & \int_0^t \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t - \phi_\epsilon - np_\epsilon) \cdot \overline{\mathbb{H}}\left(D_\tau^n - d_{\tau'} - t'\right) \mathrm{d}t' \Bigg\} \end{aligned} \tag{192}$$

$$\begin{aligned} &= \int_0^t \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \\ &\quad \cdot \min \left\{ \underline{\mathbb{H}}\left(t - t'\right), \overline{\mathbb{H}}\left(D_\tau - d_{\tau'} - t'\right) \right\} \mathrm{d}t' \end{aligned} \tag{193}$$

Because $\underline{\mathbb{H}}\left(t - t'\right) \rightarrow \{0, 1\}$ and $\overline{\mathbb{H}}\left(D_\tau - d_{\tau'} - t'\right) \rightarrow \{0, 1\}$ the function $\min(a, b)$ is equal to a logical *or* and therefore $\min(a, b) = a \cdot b$:

$$\begin{aligned} w_{\tau'}(t, D_\tau) = & \int_0^t \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \\ & \cdot \overline{\mathbb{H}}\left(D_\tau - d_{\tau'} - t'\right) \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \end{aligned} \tag{194}$$

$\overline{\mathbb{H}}\left(D_\tau - d_{\tau'} - t'\right) = 1$ if $t' - d_{\tau'} < D_\tau$. The point at which the Heaviside function switches from 1 to 0 is at $t' = \phi_{\tau'} + np_{\tau'} = D_{\tau'}$ and therefore

$$\begin{aligned} & \int_0^t \sum_{\epsilon \in \mathcal{E}_\tau} \sum_{n=0}^{k-1} \delta(t' - \phi_\epsilon - np_\epsilon) \cdot \overline{\mathbb{H}}\left(D_\tau^n - D_{\tau'}^n\right) \\ & \quad \cdot \underline{\mathbb{H}}\left(t - t'\right) \mathrm{d}t' \end{aligned} \tag{195}$$

□

Example 7 (Spreading worst case in dynamic scheduling) Let us now consider the schedules of the example task set as given in Appendix B and its schedules shown in Fig. 5 as given in detail: Figure 5a the static schedule and does not hold the deadline of task τ_3 because the task sets utilization is exact $U_\Gamma = 1$. Therefore it exists only a dynamic schedule. Related work [25,41] assumes that, if the absolute deadlines of jobs are equal, any of these jobs are scheduled. This assumption leads to different schedules as shown in Fig. 5b–d. The worst-case in this scenario is that the worst-case response time of each task is equal to its relative deadline because of the chosen utilization. This leads to the worst case response times $r_{\tau_1}^+ = 8$, $r_{\tau_2}^+ = 16$ and $r_{\tau_3}^+ = 24$. However, as seen in Fig. 5, the worst case of different tasks occur in different schedules.

The behaviour of a dynamic scheduler like EDF is non-deterministic. If no additional criterion is given, a dynamic scheduler may dispatch any of the tasks if deadlines are equal. However, this is true for static scheduling as well, and this case is prevented by giving different priorities to tasks. In the model of arbitrary deadlines, the original concept given by [32] has been expanded by [48]. In this model, any job of a task with the same priority is dispatched if its request time is shorter than any request time of other jobs with the same priority. If we add this simple criterion to dynamic scheduling, the schedule of all jobs becomes deterministic.

Theorem 12 (Spreading worst case in dynamic scheduling) *On the assumption that a dynamic scheduler is free do decide which job is scheduled if the absolute deadlines are equal the worst-case respond time of different tasks occur in different*

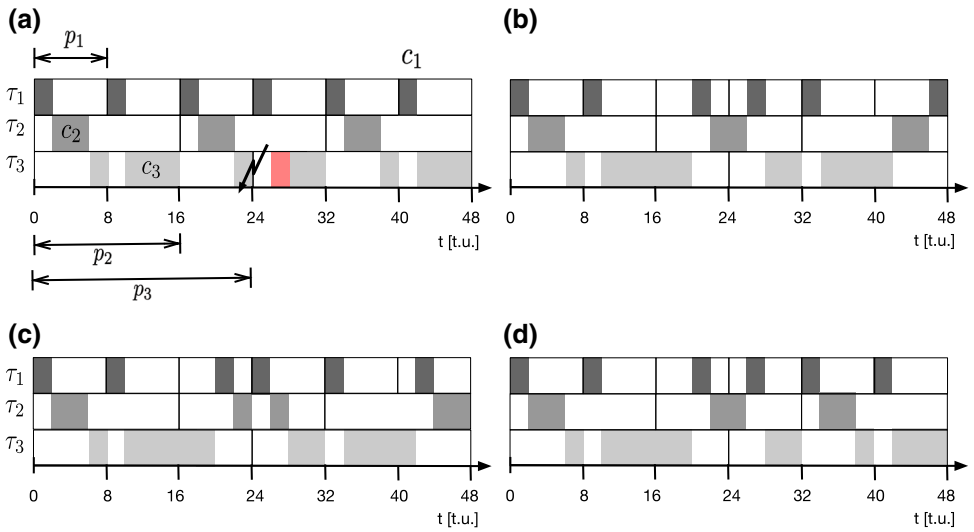


Fig. 5 Schedules of the example tasks set: in static (**a**) and dynamic scheduling (**b–d**)

schedules and therefore the worst case response time is over estimated:

$$\mathbb{E}^{n\leq k}_{\tau,\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau,\epsilon}\cdot\underline{\mathbb{H}}\left(D^n_\tau-D^n_{\tau'}\right)+\delta_{D^n_{\tau'},D^n_\tau}\cdot\overline{\mathbb{H}}\left(t^r_{\tau'}-t^r_\tau\right)\leq\mathbb{E}^{n\leq k}_{\tau,\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau,\epsilon}\cdot\overline{\mathbb{H}}\left(D^n_\tau-D^n_{\tau'}\right)\tag{196}$$

Proof If we consider the interfering request bound the proof follows directly from Corollary 2 and Theorem 11:

$$\mathbb{E}^{n\leq k}_{\tau,\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau,\epsilon}\cdot\overline{\mathbb{H}}\left(D^n_\tau-D^n_{\tau'}\right)=\mathbb{E}^{n\leq k}_{\tau,\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau,\epsilon}\cdot\underline{\mathbb{H}}\left(D^n_\tau-D^n_{\tau'}\right)+\delta_{D^n_{\tau'},D^n_\tau}\tag{197}$$

Therefore, if we add any criteria in the case that the absolute deadlines are equal, e.g. the request time of a job like $\delta_{D^n_{\tau'},D^n_\tau}\cdot\overline{\mathbb{H}}\left(t^r_{\tau'}-t^r_\tau\right)$, than in some cases no interference will occur. If an interference will not occur in some cases the resulting request is lower:

$$\mathbb{E}^{n\leq k}_{\tau,\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau,\epsilon}\cdot\underline{\mathbb{H}}\left(D^n_\tau-D^n_{\tau'}\right)+\delta_{D^n_{\tau'},D^n_\tau}\cdot\overline{\mathbb{H}}\left(t^r_{\tau'}-t^r_\tau\right)\leq\mathbb{E}^{n\leq k}_{\tau,\epsilon}(t,\Delta^b_a)\cdot C^n_{\tau,\epsilon}\cdot\overline{\mathbb{H}}\left(D^n_\tau-D^n_{\tau'}\right)\tag{198}$$

Because in periodically scheduling the critical jobs of $n-1$ task will have a request time shorter than the request time of one task, the worst case response time of all these jobs, except one, will be short compared to the situation the job can be free choose by the scheduler. □

As we have seen in by the schedules of example given in the Appendix B and shown in Fig. 5b–d. the worst-case behaviour of dynamically scheduled tasks spreads over different schedules. Theorem 12 proofs that a task set of n tasks will have shorter worst-case response bounds for $n-1$ tasks and one worst-case time equal to related work. Therefore, to the best of our knowledge, we found a closer worst-case response time estimation bound than any related work. If we only assume a dynamic schedule schedules the task with the lowest request time first, if the absolute deadlines are equal, then the worst case response time is tighter.[24] Consider Example 7 again: If we add this additional criteria to the scheduler, the maximal response times become $r^+_{\tau_1}=8$, $r^+_{\tau_2}=10$ and $r^+_{\tau_3}=20$. However, because of the presented assumptions and theorems, we can always be sure to be equal or better than previous work [25].

7 Conclusion

This paper was motivated by the question whether it exists one unified event- or request bound for all kind of analysis purposes in real-time scheduling theory. Such a function was discovered by applying mathematical techniques from theoretical physics and digital signal processing to the real-time analysis problem. It could be shown that such a unified request bound, and the definition of an average load in real-time systems allows to derivate most of the established real-time analysis algorithms from only these two assumptions. This results in an utilization based analysis and task response time analysis by just one unified event bound. Additionally, static and dynamic scheduling is considered as well in just one equation. The new equation system also covers the analysis of bursty event sequences by introducing hierarchical event streams or event densities as a computation

[24] This assumption is well accepted in static scheduling, therefore it is not surprising to adapt it to dynamic scheduling.

[25] Note, if two jobs will have the same request time and the same absolute deadline the problem arises again and another criterion must be considered. However, it is easy to add any of this to the interference mask.

Table 2 Unified real time scheduling analysis

Request bound for	
Static scheduling	Dynamic scheduling
$\mathbb{R}^{d_{\tau'} \le d_\tau}_{\tau,\epsilon}(t, \Delta^b_a) =$	$\mathbb{R}^{D^n_{\tau'} \le D^n_\tau}_{\tau,\epsilon}(t, \Delta^b_a) =$
$\mathbb{E}^{n \le k}_{\tau,\epsilon}(t, \Delta^b_a) \cdot C^n_{\tau,\epsilon} \cdot \underline{\mathbb{H}}(d_\tau - d_{\tau'})$	$\mathbb{E}^{n \le k}_{\tau,\epsilon}(t, \Delta^b_a) \cdot C^n_{\tau,\epsilon} \cdot \underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right)$
$+\delta_{d_{\tau'},d_\tau} \cdot \overline{\mathbb{H}}\left(t^r_\tau - t^r_{\tau'}\right)$	$+\delta_{D^n_{\tau'},D^n_\tau} \cdot \overline{\mathbb{H}}\left(t^r_\tau - t^r_{\tau'}\right)$
Hierarchical Scheduling	
$\mathbb{E}^{n \le k}_{\tau,\epsilon}(t, \Delta^b_a) \cdot C^n_{\tau,\epsilon} \cdot \max(\, \underline{\mathbb{H}}(\pi_{\tau'} - \pi_\tau)\,, \delta_{\tau,\tau'} \cdot [\, \underline{\mathbb{H}}\left(D^n_\tau - D^n_{\tau'}\right) + \delta_{D^n_{\tau'},D^n_\tau} \cdot \overline{\mathbb{H}}\left(t^r_\tau - t^r_{\tau'}\right)]\,)$	
Average Load	
$\mathbb{U}_\Gamma(\Delta^b_a) = \mathbb{U}_\Gamma(t) = \frac{\mathbb{R}_\Gamma(t,\infty)}{t}$	
Utilization Analysis	Response Time Analysis
$\mathbb{R}^{*\le *}_{\tau,\epsilon}(t - d_{\tau,\epsilon}, \mathcal{P}) \le t$	$\mathbb{L}^\infty_\tau(\Delta^t_0) + \mathbb{R}^{*\le *}_{\tau,\epsilon}(t,t) - t = 0$

of the convolution of two independent event densities. Additional, it is shown that the unified request bound covers the model of arbitrary deadlines as well. As a beautiful result, the work allows easily defining hierarchical schedulers. Table 2 gives an overview of the concluding results of this work. We conclude that an interfering request bound for static and dynamic scheduling for the first time in real-time scheduling theory is described by using the same equation structure! Both aspects are covered if we use, for static scheduling, the relative deadlines, and for dynamic scheduling, the absolute deadlines in the equation of interference. It could easily be seen that a few equations with a general mathematical structure will cover the main aspects in preemptive static and dynamic scheduling in the bounded execution time programming of real-time systems. In addition to these results, we also noted that the well-known response time analysis in dynamic scheduling overestimates. In the context of our new mathematical model, we found a better limit for the response time in dynamic scheduling as given in related work.

In future work, the new model is extended to the adaptive rate model. Because of the rich mathematical models are given in calculus, it should be interesting to investigate the impact of the work. First, by considering the real-time calculus to extend modular models and develop new models for modern Fieldbus devices. As the general approach of interfering request bounds builts on an abstract criterion, it should be easy to extend the work to multicriticality systems and other widely implemented scheduling algorithms such as time division multiplex (TDMA). In this paper, we have not discussed the computational complexity of the problem. The first goal was to develop a new toolbox for real-time scheduling analysis. Combining the approach of this work with the concept of [14] should be an exciting task in the future. In such work, the equation presented should be an input to the theorem proofer used in [14]. Bringing these works together will allow simple construction of any scheduler while the correctness will be proved automatically. However, the complexity of the problem is exponential. Therefore approximation techniques already discussed has to be integrated into future work.

Acknowledgements This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grand SL 47/17-1. We gratefully thank Chekib Khezami to simplify the proof of the remaining load, Iwan Feras Fattohi for his critical comments and Kilian Kempf for proofreading. Finally, the paper is dedicated to Ulrich. First of all, to Frank Slomkas dad, Cpt. Ulrich Slomka, who teaches me the curiosity to the world, second to Prof. Ulrich Herzog who helped Frank Slomka to tame this curiosity and to give my work a scientific structure.

Funding Open Access funding enabled and organized by Projekt DEAL.

A Mathematical framework

The main focus of the paper is the adaption of the mathematics used in theoretical physics and digital signal theory. In this appendix, we explain a few notations which are typically not well-known or widely used in the real-time systems community. Additionally, a list of symbols clarifies the notation. One of the goals of this paper is to formulate an easy to use mathematical theory of real-time systems with a clear focus on intuitively simple equations. Therefore a table which lists all symbols is given at the end of the appendix.

For intuitive reading we write p_τ. This means a function which gives the period of the specified task; The idea is a short notation for $p_\tau = p(\tau)$. The next two definitions are from theoretical physics. In the real-time analysis, we often write summations, and in this paper, we get integrals over two summations. However, writing this in each equation brings a lot of overhead and redundant information. Therefore, we adopt an index based writing notation to the problem:

Definition 23 (*Einstein's notation*)

$$i \in \{1, ..., n\}: \qquad c_i x^i = c_1 x^1 + ... + c_3 x^3 = \sum_{i=1}^{n} c_i x^i \tag{199}$$

The notation was introduced by [20] to simplify multidimensional equations in gravity. However, it can be used to simplify the notations in real-time analysis as well. In this work we use two modified forms to reduce the complexity of equations:

Definition 24 (*Modified Einstein's Notation*)

$$c_i^k x_i = c_1 x_1 + \cdots + c_k x_k = \sum_{i=1}^{k} c_i x_i \tag{200}$$

$$c_{i,j}^k x_{i,j} = \sum_{i=1}^{k} \sum_{j \in J} k \cdot c_{i,j} x_{i,j} \tag{201}$$

This idea can be adapted to the request bound:

$$\begin{aligned} &\int_{-\infty}^{\infty} \delta(t' - \phi_\epsilon - np_\epsilon)_{\tau,\epsilon}^{n \le k} \cdot C_{\tau,\epsilon}[n \bmod |C|] \\ &\quad \cdot \overline{\mathbb{H}}\left(t' - t_a\right) \cdot \underline{\mathbb{H}}\left(t_b - t'\right) \, \mathrm{d}t' \\ &= \int_{[a,b)} \delta(t' - \phi_\epsilon - np_\epsilon)_{\tau,\epsilon}^{n \le k} \cdot C_{\tau,\epsilon}[n \bmod |C|] \cdot \mathbb{S}_{\tau',\phi_\epsilon + np_\epsilon}^{\tau,\epsilon}, \, \mathrm{d}t' \\ &= \mathbb{E}_{\tau,\epsilon}^{n \le k}(t, \Delta_a^b) \cdot C_{\tau,\epsilon}^n \cdot \mathbb{S}_{\tau,\epsilon,\tau'}^n = \mathbb{R}_{\tau,\epsilon}^{n \le k}(t, \Delta_a^b) \cdot \mathbb{S}_{\tau,\epsilon,\tau'}^n \end{aligned}$$

An other useful symbol is the Kronecker delta. This function only returns 1 if both arguments are equal. In the other case the result is 0. It is used in physics as a short hand notation for matrices. In our case it is used to collect tasks of the same priority. It also can be used to collect jobs of the same tasks.

Definition 25 (*Kronecker Delta*) The Kronecker delta is used to write matrices in a compact form. The function returns a 1 if the two elements given to the function are equal. In all other cases it returns 0. In real-time analysis this property can be used to identify tasks with the same priority:

$$\delta_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{202}$$

The following table concludes all symbols used in the paper. The first part of the table gives the functions applied from theoretical physics. The second part list all time-related symbols, while the third part introduces event-related elements. The fourth part presents all parameters related to tasks, and the last part lists the symbols used in real-time analysis.

Symbol	Meaning
$\delta(t)$	Dirac delta
$\delta_{i,j}$	Kronecker delta
$\mathbb{H}(t)$	Heaviside function
$\overline{\mathbb{H}}(t)$	Upper Heaviside function
$\underline{\mathbb{H}}(t)$	Lower Heaviside function
t_a	Point in time
$t_{\epsilon'}$	Request time of an event
Δ	Time interval
Δ^+	Maximal time interval
Δ^-	Minimal time interval
Δ_0	Interval related to time 0, equivalent to t_0
Δ_a^b	Interval between t_a and t_b
p	Period
j	Jitter
ϕ	Minimal distance, phase or offset
$\mathcal{P}$	Hyper period, the least common multiplier of a set of periods
$\mathcal{P}_\Gamma$	Hyper period of a task set
ϵ'	Event
ϵ	Event tuple describing a periodic event sequence
$\mathcal{E}$	List of event tuples
m	Event sequence of an event list
m_Δ	Event stream or event density
m_Δ^+	Maximal event density
m_Δ^-	Minimal event density
τ, τ_n	Task
$\tau_{n,\epsilon}$	Job
τ'	Interfering task
$\tau_{,}'$	Interfering job
Γ	Task set
π	Priority
$\overline{\Gamma}_\tau$	Higher priority task set of task τ
d	Relative deadline
D	Absolute deadline $D = p + d$
c	Computational load, execution time
c^-	Best case execution time
c^+	Worst case execution time
$C_{\tau,\epsilon}^+$	A vector of different worst case execution times
$C_{\tau,\epsilon}^-$	A vector of different best case execution times
$\mathcal{D}$	A vector of different relative deadlines
$\mathbb{S}_{\tau,\tau'}^\square$	Scheduler, schedules jobs of tasks τ, τ' according to criterion $\square$
$\mathbb{E}$	Event bound function
$\mathbb{E}_\tau$	Event bound function of a task
$\mathbb{E}_\Gamma$	Event bound function of a task set
$\mathbb{R}$	Request bound function
$\mathbb{D}$	Demand bound function
$\mathbb{L}$	Remaining load
r_τ	Response time of a task
r_τ^+	Maximal response time of a task
r_τ^-	Minimal response time of a task
U_τ	Utilization of a task
U_Γ	Utilization of a task set

B Examples

We use a computer algebra system (CAS) [17] to validate the approach. The CAS allows us to verify the algebraic structure of the work. Additionally, it is possible to consider numeric examples as well. Table 3 gives a task set used as a running example in the rest of the paper. Just for simplification, we only consider periodic tasks. Therefore we specify three tasks by their period, their worst-case execution time and the relative deadline which is given by the deadline as well. Additionally, the last column of the table states the input for the computer algebra system, as mentioned earlier. Note that this task set has a utilization equal to one. Therefore it is schedulable by dynamic scheduling and not by static scheduling as shown later in Fig. 5. Considering a utilization of one is essential to investigate the differences in static and dynamic scheduling and the tightness of a response-time analysis as seen later. Let us first compute the $EventDensity$ following Definition 12 by the CAS to build the algebraic equation from a given nested list as task description:

```
In[ ]:= Task1 := {{{{0, 8}, Infinity}}, 2}}
        Task2 := {{{{0, 16}, Infinity}}, 4}}
        Task3 := {{{{0, 24}, Infinity}}, 12}}
        TaskSet := {Task1, Task2, Task3}

In[ ]:= EventDensity[Task1, td] + EventDensity[Task2, td] + EventDensity[Task3, td]

Out[ ]= ∑_{IoE=0}^{∞} DiracDelta[-24 IoE + td] + ∑_{IoE=0}^{∞} DiracDelta[-16 IoE + td] + ∑_{IoE=0}^{∞} DiracDelta[-8 IoE + td]
```

The variable IoE (Instance of Event) denotes the number of the considered job and the function $EventDensity$ computes the Dirac comp as discussed earlier. Replacing it by n or any other counting variable leads to the formal notation given earlier. Note that the sequence of variables given in the CAS output follows the rules defined in computer algebra. Therefore we do not change outputs of the CAS to be compatible with the equations defined. Assume now we defined a function $DiracCount$ to count events. Then the CAS gives tho following output if we like to count the events in the interval $[0, T)$:

```
In[•]:= DiracCount[{EventDensity[Task1, td], td}, {0, T}, t] +
        DiracCount[{EventDensity[Task2, td], td}, {0, T}, t] +
        DiracCount[{EventDensity[Task3, td], td}, {0, T}, t]

Out[•]= ∫_{-∞}^{t} HeavisideDown[T - td] × HeavisideUp[td] ∑_{IoE=0}^{∞} DiracDelta[-24 IoE + td] dtd +
        ∫_{-∞}^{t} HeavisideDown[T - td] × HeavisideUp[td] ∑_{IoE=0}^{∞} DiracDelta[-16 IoE + td] dtd +
        ∫_{-∞}^{t} HeavisideDown[T - td] × HeavisideUp[td] ∑_{IoE=0}^{∞} DiracDelta[-8 IoE + td] dtd
```

The interfering request bounds in static and dynamic scheduling are given in Fig. 6. In this example, we consider the interfering request bounds of the two jobs $\tau_{1,2}$ and task $\tau_{2,2}$. Because only two jobs during the hyper-period occur from task τ_3, we consider the interfering request bound of the second job $\tau_{3,1}$. Note that it is possible to compute the interference of each job of each task. However, we chose the example jobs because the difference between static and dynamic scheduling is easily seen. The following CAS input produces the resulting graphs of 6:

Table 3 Example task set

Task	Period $p\ [t.u.]$	Wcet $c^+\ [t.u.]$	Relative deadline $d\ [t.u.]$	CAS input
τ_1	8	2	8	$\{\{\{\{0, 8\}, Infinity\}\}, 2\}\}$
τ_2	16	4	16	$\{\{\{\{0, 16\}, Infinity\}\}, 4\}\}$
τ_3	24	12	24	$\{\{\{\{0, 24\}, Infinity\}\}, 12\}\}$

```
In[•]:= Plot[{
         RequestBoundN[{Task1[2]}, {TaskSet, dms}, {0, t}, t],
         RequestBoundN[{Task2[2]}, {TaskSet, dms}, {0, t}, t],
         RequestBoundN[{Task3[1]}, {TaskSet, dms}, {0, t}, t]}, {t, 0, Hyperperiod[TaskSet]},
       AxesOrigin → {0, 0}, PlotLegends → Placed["Expressions", Below], AxesLabel → {t, R}]
In[•]:= Plot[{
         RequestBoundN[{Task1[2]}, {TaskSet, edf}, {0, t}, t],
         RequestBoundN[{Task2[2]}, {TaskSet, edf}, {0, t}, t],
         RequestBoundN[{Task3[1]}, {TaskSet, edf}, {0, t}, t]}, {t, 0, Hyperperiod[TaskSet]},
       AxesOrigin → {0, 0}, PlotLegends → Placed["Expressions", Below], AxesLabel → {t, R}]
```

Let us compute the remaining load of job 3 for a static and dynamic scheduler by the following CAS input. The resulting plot is shown on the left hand in Fig. 7. Additionally, we consider as an example, the remaining load of task τ_3, job 1. Figure 7 shows the result for static and dynamic scheduling.

```
Plot[
 {RemainingLoad[{Task1[2]}, {TaskSet, edf}, t],
  RemainingLoad[{Task1[2]}, {TaskSet, dms}, t]}, {t, 0, Hyperperiod[TaskSet] + 2},
 PlotRange → {{0, Hyperperiod[TaskSet]}, {0, 20}}, AxesOrigin → {0, 0}, PlotLegends → Placed["Expressions", Above],
 AxesLabel → {t, L}]
Plot[
 {RemainingLoad[{Task3[0]}, {TaskSet, edf}, t],
  RemainingLoad[{Task3[0]}, {TaskSet, dms}, t]}, {t, 0, Hyperperiod[TaskSet] + 2},
 PlotRange → {{0, Hyperperiod[TaskSet]}, {0, 20}}, AxesOrigin → {0, 0}, PlotLegends → Placed["Expressions", Above],
 AxesLabel → {t, L}]
```

The average load of job 1 and job 2 of task $\tau 3$ are given in Fig. 6. Note that this diagram clearly shows the busy window of both jobs. The following CAS input produces the plots (Fig. 8):

```
In[•]:= Plot[{
         Utilization[{Task3[0]}, {TaskSet, edf}, t],
         Utilization[{Task3[0]}, {TaskSet, dms}, t], 1}, {t, 0, Hyperperiod[TaskSet]}, AxesOrigin → {0, 0},
       PlotLegends → Placed["Expressions", Below], AxesLabel → {t, U}]
In[•]:= Plot[{
         Utilization[{Task3[1]}, {TaskSet, edf}, t],
         Utilization[{Task3[1]}, {TaskSet, dms}, t], 1}, {t, 0, Hyperperiod[TaskSet]}, AxesOrigin → {0, 0},
       PlotLegends → Placed["Expressions", Below], AxesLabel → {t, U}]
```

The response time analysis implemented in the CAS supports static and dynamic scheduling. The CAS gives the following output for the example task set. The output is printed as a list with the following format: $\{t^r_{\tau,\epsilon}, \mathbb{L}^{\tau' \geq \tau}_{\tau,\tau'}(t), r_{\tau,\epsilon}, d_\tau\}$. In the following each of this lists represent a task and the analysis provides the response times of each job.

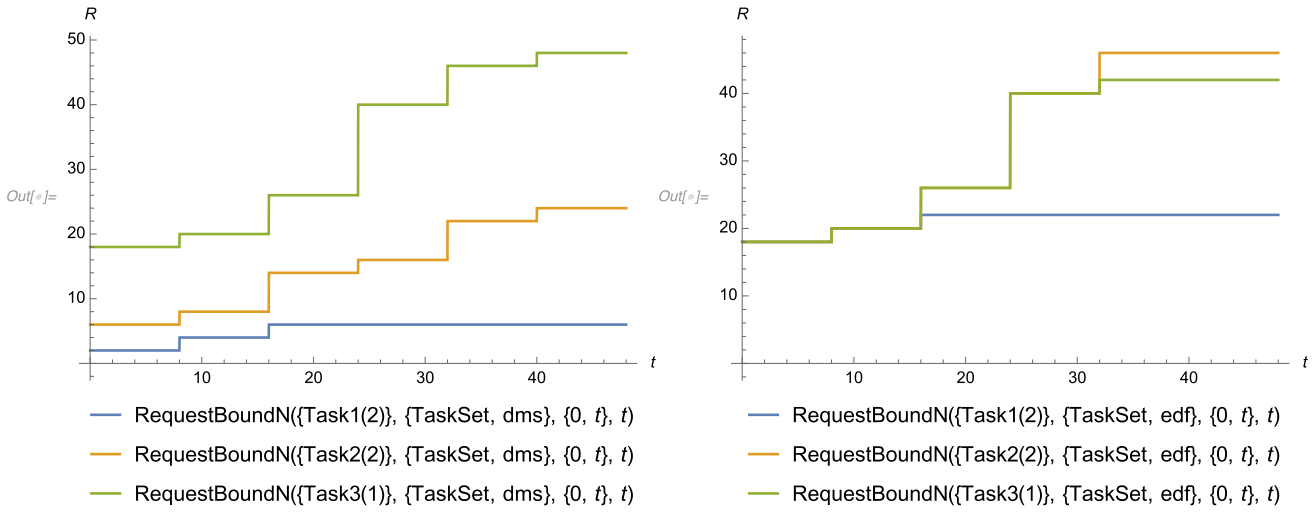


Fig. 6 Intefering request bound of the example task set on selected jobs

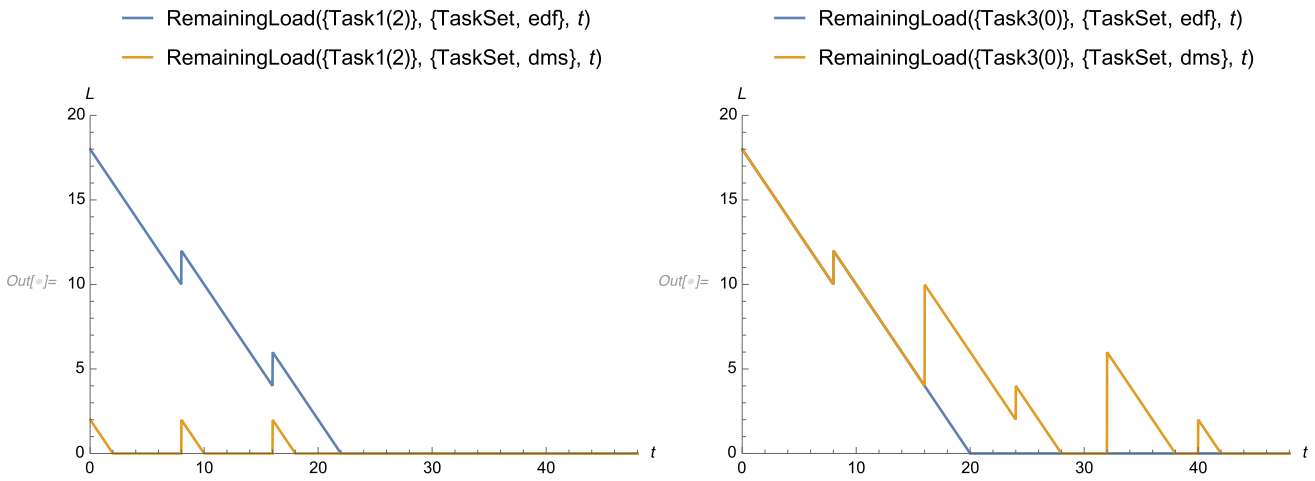


Fig. 7 Remaining load of the third job $\tau_{1,2}$ and the first job $\tau_{3,0}$

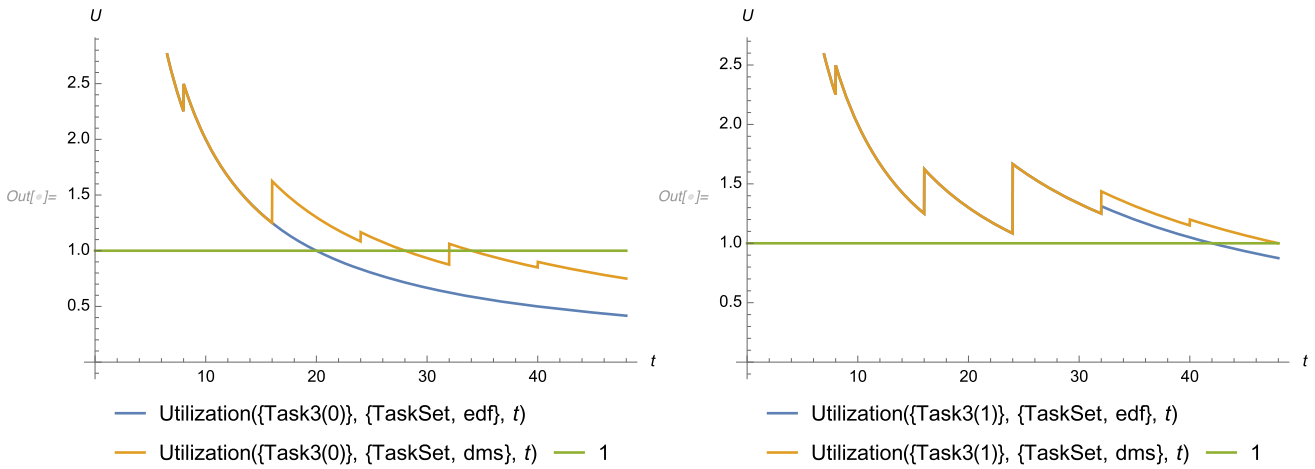


Fig. 8 Average load of first job $\tau_{3,0}$ and the second job $\tau_{3,1}$. It is easily seen when the utilization condition fulfills

```
In[•]:= ResponseTimeAnalysis[{TaskSet, dms}]

Out[•]= {{{0, 0, 2, 8}, {8, 0., 2., 8}, {16, 0., 2., 8},
          {24, 0., 2., 8}, {32, 0., 2., 8}, {40, 0., 2., 8}},
         {{0, 0, 6, 16}, {16, 0., 6., 16}, {32, 0., 6., 16}}, {{0, 0, 28, 24}, {24, 2, 24, 24}}}
In[•]:= ResponseTimeAnalysis[{TaskSet, edf}]

Out[•]= {{{0, 0, 2, 8}, {8, 0., 2., 8}, {16, 4, 6, 8}, {24, 2, 4, 8}, {32, 0., 2., 8}, {40, 6, 8, 8}},
         {{0, 0, 6, 16}, {16, 4, 10, 16}, {32, 8, 14, 16}}, {{0, 0, 20, 24}, {24, 2, 18, 24}}}
```

Based on this output, it is possible to build an intuitive plot showing the response times of all tasks instances or jobs as bars on their release time. In such a diagram, an orange plot bar indicates the computed response time at the specified release time. Negative blue bars give the remaining load at the release time as well. Additionally, in the plots given in Figs. 9 and 10 the relative deadline is given as a lightweight orange colour in the background.[26] Note that the results are the same as expected from the schedules given in Fig. 5.

To consider hierarchical scheduling, the task set given in Table 3 is modified. To highlight the effect of hierarchical scheduling, we add a few tasks and to decrease the utilization of the original task set. Some other parameters are changed as well. We therefore use the following task set:

```
In[•]:= Task1 := {{{{0, 6}, Infinity}}, 1}, 1}
        Task2 := {{{{0, 9}, Infinity}}, 1, 8}, 2}
        Task3 := {{{{0, 12}, Infinity}}, 2}, 2}
        Task4 := {{{{0, 18}, Infinity}}, 3, 15}, 2}
        Task5 := {{{{0, 36}, Infinity}}, 7, 17}, 3}
        Task6 := {{{{0, 72}, Infinity}}, 9, 35}, 4}
        TaskSet := {Task1, Task2, Task3, Task4, Task5, Task6}
```

In this task set, the last number denotes the priority level, ignored under dynamic scheduling. In static scheduling, a task with the lowest number has the highest priority. The CAS computes the following output, plotted in Figs. 11 and 12:

```
In[ ]:= ResponseTimeAnalysis[{TaskSet, dms}]

Out[ ]= {{{0, 0, 1, 6}, {6, 0., 1., 6}, {12, 0., 1., 6}, {18, 0., 1., 6},
          {24, 0., 1., 6}, {30, 0., 1., 6}, {36, 0., 1., 6}, {42, 0., 1., 6},
          {48, 0., 1., 6}, {54, 0., 1., 6}, {60, 0., 1., 6}, {66, 0., 1., 6}},
         {{0, 0, 2, 8}, {9, 0., 1., 8}, {18, 0., 2., 8}, {27, 0., 1., 8}, {36, 0., 2., 8},
          {45, 0., 1., 8}, {54, 0., 2., 8}, {63, 0., 1., 8}}, {{0, 0, 4, 12}, {12, 0., 3., 12},
          {24, 0., 3., 12}, {36, 0., 4., 12}, {48, 0., 3., 12}, {60, 0., 3., 12}},
         {{0, 0, 8, 15}, {18, 0., 5., 15}, {36, 0., 8., 15}, {54, 0., 5., 15}},
         {{0, 0, 24, 17}, {36, -5.46118×10^-9, 24., 17}}, {{0, 0, 52, 35}}}
In[ ]:= ResponseTimeAnalysis[{TaskSet, edf}]

Out[ ]= {{{0, 0, 1, 6}, {6, 0., 1., 6}, {12, 4, 5, 6}, {18, 1, 2, 6}, {24, 0., 1., 6}, {30, 7, 8, 6},
          {36, 2, 3, 6}, {42, 0, 1, 6}, {48, 6, 7, 6}, {54, 3, 4, 6}, {60, 0., 1., 6}, {66, 0, 1, 6}},
         {{0, 0, 2, 8}, {9, 6, 7, 8}, {18, 1, 3, 8}, {27, 7, 8, 8}, {36, 2, 4, 8},
          {45, 8, 9, 8}, {54, 3, 5, 8}, {63, 0, 1, 8}}, {{0, 0, 4, 12}, {12, 4, 7, 12},
          {24, 9, 13, 12}, {36, 2, 6, 12}, {48, 6, 9, 12}, {60, 2, 6, 12}},
         {{0, 0, 8, 15}, {18, 1, 6, 15}, {36, 2, 10, 15}, {54, 3, 9, 15}},
         {{0, 0, 15, 17}, {36, 2, 17, 17}}, {{0, 0, 34, 35}}}
```

As a result, the schedule of the second task set is feasible if all tasks scheduled dynamically. If a hierarchical scheduler is used then the worst-case response time of all jobs of task τ_1 is decreased because of its high priority, while task τ_6 does not hold its deadline anymore, because of its low priority. All other tasks except task $\tau 5$ have the same priority an, therefore, are scheduled dynamically. However, task τ_2 and task τ_3 will miss their deadlines because of the high priority of task τ_1.

[26] The plots shown are originally given from the CAS.

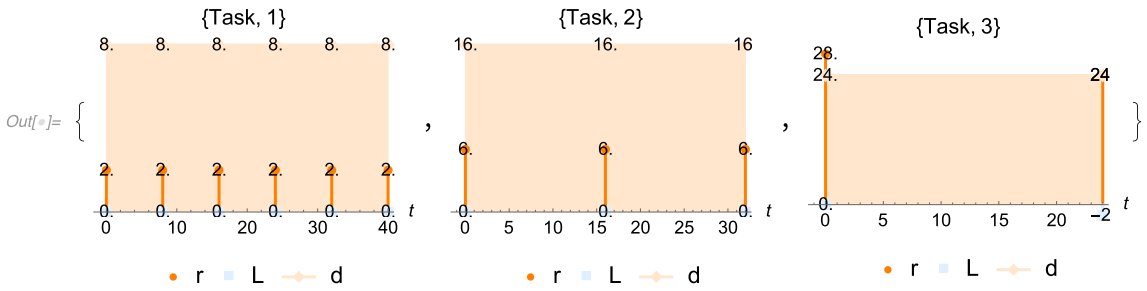


Fig. 9 Response time analysis plot of the example task set scheduled under DMS

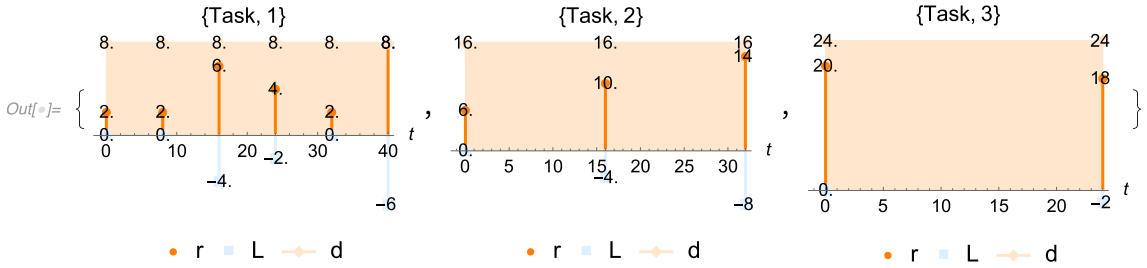


Fig. 10 Response time analysis plot of the example task set scheduled under EDF

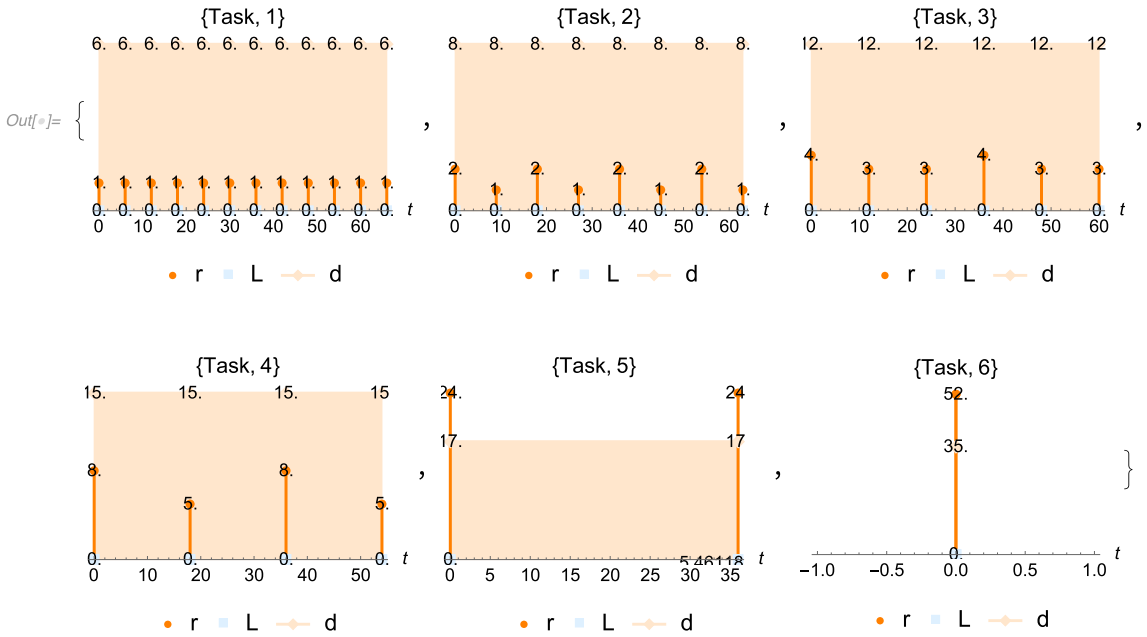


Fig. 11 Response time analysis plot of the second task set scheduled hierarchical by DMS and EDF

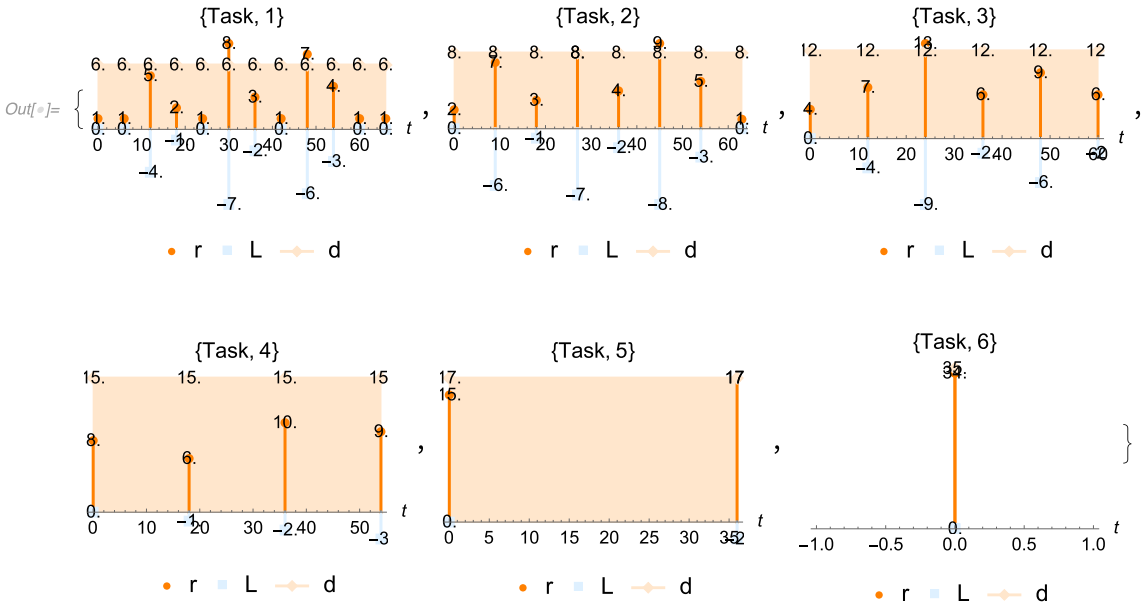


Fig. 12 Response time analysis plot of the second task set scheduled only by EDF

References

1. Albers K, Bodmann F, Slomka F (2006) Hierarchical event streams and event dependency graphs: a new computational model for embedded real-time systems. In: 18th Euromicro conference on real-time systems, ECRTS'06, 5–7 July 2006, Dresden, Germany, Proceedings, pp 97–106
2. Albers K, Bodmann F, Slomka F (2008) Advanced hierachical event-stream model. In: Euromicro conference on real-time systems, 2008. In: ECRTS'08. IEEE, pp 211–220
3. Albers K, Slomka F (2004) An event stream driven approximation for the analysis of real-time systems. In: 16th Euromicro conference on real-time systems (ECRTS 2004), 30 June–2 July 1004, Catania, Italy, proceedings, pp 187–195
4. Albers K, Slomka F (2005) Efficient bound tests analysis for real-time systems with EDF scheduling. In: 2005 Design, automation and test in Europe conference and exposition (DATE 2005), 7–11 March 2005. Munich, Germany, pp 492–497
5. Anssi S, Kuntz S, Gérard S, Terrier F (2013) On the gap between schedulability tests and an automotive task model. J Syst Architect 59(6):341–350
6. Baruah S (2003) Dynamic- and static-priority scheduling of recurring real-time tasks. Real Time Syst 24:93–128. https://doi.org/10.1023/A:1021711220939
7. Baruah S (2010) The non-cyclic recurring real-time task model. In: 2010 IEEE 31st real-time systems symposium (RTSS), pp 173–182
8. Baruah S, Chen D, Gorinsky S, Mok A (1999) Generalized multiframe tasks. Real Time Syst 17:5–22. https://doi.org/10.1023/A:1008030427220
9. Baruah S, Howell R, Rosier L (1993) bound tests problems for recurring tasks on one processor. Theor Comput Sci 118(1):3–20
10. Baruah S, Mok A, Rosier L (1991) Preemptively scheduling hard-real-time sporadic tasks on one processor. In: 90 Real-time system on symposium. IEEE, pp 182–190
11. Baruah SK, Bonifaci V, d'Angelo G, Marchetti-Spaccamela A, Van Der Ster S, Stougie L (2011) Mixed-criticality scheduling of sporadic task systems. In: European symposium on algorithms, pp 555–566. Springer
12. Biondi A, Melani A, Marinoni M, Natale MD, Buttazzo G (2014) Exact interference of adaptive variable-rate tasks under fixed-priority scheduling. In: Proceedings of the 26th Euromicro conference on real-time systems (ECRTS)
13. Boudec J-YL (1998) Application of network calculus to guaranteed service networks. IEEE Trans Inf Theory 44(3):1087–1096
14. Bozhko S, Brandenburg BB (2020) Abstract response-time analysis: a formal foundation for the busy-window principle. In: 32nd Euromicro conference on real-time systems (ECRTS 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik
15. Bracewell R (2000) The Fourier transform and its applications. Electrical engineering series. McGraw Hill, New York
16. Chakraborty S, Künzli S, Thiele L (2003) A general framework for analysing system properties in platform-based embedded system designs. In: Design, automation and test in Europe conference and exhibition 2003, pp 190–195
17. Cohen JS (2003) Computer algebra and symbolic computation: mathematical methods. CRC Press, Boca Raton
18. Cruz RL (1991) A calculus for network delay. i. Network elements in isolation. IEEE Trans Inf Theory 37(1):114–131
19. Cruz RL (1991) A calculus for network delay. ii. network analysis. IEEE Trans Inf Theory 37(1):132–141
20. Einstein A (1916) The foundation of the general theory of relativity. Princton University Press, Princton
21. Fisher N, Baruah S (2005) A polynomial-time approximation scheme for bound tests analysis in static-priority systems with bounded relative deadlines. In: Proceedings of the 13th international conference on real-time systems
22. Graham R (1976) Bounds on the performance of scheduling algorithms. Comput Job Sched Theory, pp 165–227
23. Gresser K (1993) An event model for deadline verification of hard real-time systems. In: Fifth Euromicro workshop on real-time systems, 1993. Proceedings. IEEE, pp 118–123
24. Guan N, Yi W (2014a) General and efficient response time analysis for edf scheduling. In: Design, automation & test in Europe conference & exhibition (DATE)
25. Guan N, Yi W (2014b) General and efficient response time analysis for edf scheduling. In: Proceedings of the conference on design, automation & test in Europe, DATE '14, pp 255:1–255:6, 3001

Leuven, Belgium, Belgium. European Design and Automation Association
26. Ittershagen P, Hartmann PA, Grüttner K, Rettberg A (2013) Hierarchical real-time scheduling in the multi-core era-an overview. In: 2013 IEEE 16th international symposium on object/component/service-oriented real-time distributed computing (ISORC). IEEE, pp 1–10
27. Joseph M, Pandya P (1986) Finding response times in a real-time system. Comput J 29(5):390–395
28. Künzli S, Hamann A, Ernst R, Thiele L (2007) Combined approach to system level performance analysis of embedded systems. In: Proceedings of the 5th IEEE/ACM international conference on hardware/software codesign and system synthesis, CODES+ISSS '07. ACM, New York, NY, USA, pp 63–68
29. Lehoczky JP (1990) Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: 11th real-time systems symposium, 1990. Proceedings, pp 201 –209
30. Leung JY-T, Whitehead J (1982) On the complexity of fixed-priority scheduling of periodic, real-time tasks. Perform Eval 2(4):237–250
31. Lipari G, Bini E (2005) A methodology for designing hierarchical scheduling systems. J Embed Comput 1(2):257–269
32. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. J ACM 20:46–61
33. Mindell D (2008) Digital Apollo: human and machine in spaceflight. Inside technology series. MIT Press, Cambridge
34. Mok AK, Chen D (1997) A multiframe model for real-time tasks. IEEE Trans Softw Eng 23(10):635–645
35. Moyo NT, Nicollet E, Lafaye F, Moy C (2010) On schedulability analysis of non-cyclic generalized multiframe tasks. In: 2010 22nd Euromicro conference on Real-time systems (ECRTS), pp 271 –278
36. Naedele M, Thiele L, Eisenring M (1998) Characterising variable task releases and processor capacities. Technical Report 45, Computer Engineering and Networks Laboratory, ETH Zurich
37. Palencia JC, Harbour MG (1998) Schedulability analysis for tasks with static and dynamic offsets. In: The 19th IEEE real-time systems symposium, 1998. Proceedings, pp 26 –37
38. Palencia JC, Harbour MG (2003) Offset-based response time analysis of distributed systems scheduled under edf. In: 15th Euromicro conference on real-time systems proceedings of the proceedings
39. Palencia JC, Harbour MG (2005) Response time analysis of edf distributed real-time systems. In: J Embed Comput, Nr. 2
40. Richter K (2005) Compositional scheduling analysis using standars event models. Ph.D. thesis, TU Braunschweig
41. Spuri M (1996) Analysis of deadline scheduled real-time systems. Ph.D. thesis, Inria
42. Stigge M, Ekberg P, Guan N, Yi W (2011) The digraph real-time task model. In: 2011 17th IEEE real-time and embedded technology and applications symposium (RTAS), pp 71 –80
43. Stigge M, Yi W (2013) Combinatorial abstraction refinement for bound tests analysis. In: Proceedings of the 34th IEEE real-time systems symposium (RTSS)
44. Thiele L, Chakraborty S, Gries M, Künzli S (2002) A framework for evaluating design tradeoffs in packet processing architectures. In: 39th design automation conference, 2002. Proceedings, pp 880–885
45. Thiele L, Chakraborty S, Gries M, Maxiaguine A, Greutert J (2001) Embedded software in network processors—models and algorithms. In: Henzinger T, Kirsch C (eds) Embedded software, volume of 2211 lecture notes in computer science. Springer, Berlin, pp 416–434
46. Thiele L, Chakraborty S, Naedele M (2000) Real-time calculus for scheduling hard real-time systems. In: International symposium on circuits and systems ISCAS (2000) vol 4. Switzerland, Geneva, pp 101–104
47. Tindell KW, Burns A, Wellings AJ (1994) An extendible approach for analyzing fixed priority hard real-time tasks. Real Time Syst 6:133–151. https://doi.org/10.1007/BF01088593
48. Tindell KW, Burns A, Wellings AJ (1994) An extendible approach for analyzing fixed priority hard real-time tasks. Real Time Syst 6(2):133–151
49. Tindell KW, Clark J (1994) Holistic schedulability analysis for distributed hard real-time systems. Microprocess Microprogram 40(2–3):117–134
50. Wandeler E, Thiele L (2006) Interface-based design of real-time systems with hierarchical scheduling. In: 12th IEEE real-time and embedded technology and applications symposium (RTAS'06). IEEE, pp 243–252
51. W.Tindell K (1994) Adding time-offsets to schedulability analysis. Technical report, Department of Computer Science, University of York
52. Zhu H, Goddard S, Dwyer MB (2011) Response time analysis of hierarchical scheduling: the synchronized deferrable servers approach. In: 2011 IEEE 32nd real-time systems symposium (RTSS). IEEE, pp 239–248

Frank Slomka received the Dipl.-Ing. degree in electrical engineering with the Technical University of Braunschweig, Germany, in 1993 and the Ph.D. degree from the University of Erlangen–Nuremberg, Germany, in 2002. He was with Bosch Telecom, Germany, from 1993 to 1996, where he was responsible for the development of system software for digital wireless telephones. From 1996 to 2001, he worked on a research project on hardware/software codesign of communication systems with the University of Erlangen–Nuremberg. In 2002, he founded the company INCHRON, Germany, together with some colleagues from Erlangen. The company deals with all aspects of real-time analysis. Since then he has been a Technical Consultant with INCHRON. From 2002 to 2007, he was an Assistant Professor of embedded systems with the University of Oldenburg, Germany, and since 2007, he has been a Full Professor of embedded and real-time systems with Ulm University, Ulm, Germany.

Mohammadreza Sadeghi received his M.Sc. degree in communication technology in 2016 from the Ulm University, Germany. He is currently working with INCHRON AG, Erlangen, Germany and pursuing his Ph.D. degree at the Department of Embedded/Real-Time Systems, Ulm University, Germany. His current research interests include real-time analysis of systems running adaptive variable-rate tasks.