State-Based Mode Switching with Applications to Mixed-Criticality Systems

Pontus Ekberg, Martin Stigge, Nan Guan and Wang Yi
Uppsala University, Sweden
Email: {pontus.ekberg | martin.stigge | nan.guan | yi}@it.uu.se

***Abstract*—We present a new graph-based real-time task model that can specify complex job arrival patterns and global state-based mode switching. The mode switching is of a mixed-criticality style, enabling immediate changes to the parameters of active jobs upon mode switches. The resulting task model therefore generalizes previously proposed task graph models as well as mixed-criticality (sporadic) task models, and further allows for the modeling of timing properties not found in any of these models. We outline an EDF uniprocessor schedulability analysis procedure by combining ideas from prior analysis methods for graph-based and mixed-criticality task scheduling.**

I. Introduction

We present the Mode-Switching Digraph Real-Time (MS-DRT) task model, a model that can express complex arrival patterns of jobs and global mode switching. The tasks are represented with graphs that specify both the arrival patterns and the synchronization points (mode switches) between tasks. MS-DRT is a strict generalization of the Digraph Real-Time (DRT) task model [1] and of the common mixed-criticality sporadic task model [2], its variations [3] and generalizations [4]. MS-DRT also enables the modeling of many timing properties that have no counterpart in the above models, some examples of which are shown in Section III.

The modes in MS-DRT are system-wide, meaning that all tasks synchronously switch from one mode to another. Mode-switching logic is specified per state (vertex) of the task graphs, so that behaviors may differ depending on the local state of the tasks. The mode change protocol is of a generalized mixed-criticality style, enabling immediate changes to the timing parameters of active jobs at mode changes. As opposed to the usual mixed-criticality setting, it is possible to have cyclic mode changes in MS-DRT. In addition to being a mixed-criticality task model, MS-DRT could, for example, find applications as a timing model for statecharts [5] by considering the orthogonal components as tasks and expressing their arrival and synchronization patterns.

We outline an EDF schedulability analysis procedure for MS-DRT task systems on uniprocessors. The analysis procedure combines ideas from previously published EDF schedulability analysis methods for DRT task systems [1] and mixed-criticality sporadic task systems [6], [4]. These are all based on computing demand bound functions for tasks, and are therefore possible to combine.

A. Related Work

Mixed-criticality scheduling theory has seen a process of slowly generalized task models. After the seminal paper by Vestal [2], which described fixed-priority response-time analysis for mixed-criticality sporadic task systems, the initial research effort was based on scheduling static sequences of mixed-criticality jobs. The work by Baruah et al. [7] provides a good overview of such mixed-criticality job scheduling.

One of the scheduling theories developed for static job sequences, the OCBP scheduling approach, was then generalized for sporadic tasks systems by Li and Baruah [8]. Shortly thereafter, Baruah et al. developed a new EDF-based scheduling algorithm, called EDF-VD [9], for mixed-criticality sporadic tasks. The initial work by Vestal on response-time analysis of fixed-priority scheduling was also improved by Baruah et al. [10]. This list is by no means exhaustive, many other works have since been based on the mixed-criticality sporadic task model.

EDF-based scheduling of mixed-criticality sporadic tasks was further investigated by Ekberg and Yi [6]. This was based on very similar runtime scheduling as the previous work on EDF-VD by Baruah et al. [9], but the analysis was based on computing demand bound functions for the mixed-criticality tasks. Demand bound functions offer a handy abstraction for use in EDF-based schedulability analysis, and have been successfully applied to many varying task models outside of the mixed-criticality setting. For example, EDF scheduling analyses based on demand bound functions have been presented for task models that offer greater expressiveness than sporadic tasks regarding job arrival patterns, such as the GMF [11] and DRT [1] task models. This wide applicability of demand bound functions is what allows us analyze a combination of mixed-criticality style mode switching with more general task models for this paper.

Baruah [3] has also proposed a variation of the standard mixed-criticality sporadic task model, in which the periods of sporadic tasks rather than their execution-time estimates are subject to uncertainties. A generalization by Ekberg and Yi [4] covers the case where all parameters of the sporadic tasks may change, and the potential mode switches can be expressed as a directed acyclic graph instead of being linearly ordered. Contrary, the MS-DRT task model proposed in this paper is not constrained to sporadic behavior, and allows cyclic mode switches.

II. The Mode-Switching Digraph Real-Time Task Model

In this section we describe the syntax and semantics of the MS-DRT task model in as abstract terms as possible. Concrete examples of MS-DRT tasks, outlining some more real-world interpretations of the semantics, are presented in Section III.

A. Syntax

An MS-DRT task system is formally defined by a set of tasks $T = \{\tau_1, \ldots, \tau_n\}$ together with an associated set of modes $M(T) = \{\mu_1, \ldots, \mu_k\}$. An MS-DRT task $\tau \in T$ is defined by an ordered tuple $(V(\tau), E_{\text{cf}}(\tau), E_{\text{ms}}(\tau))$, where

- $V(\tau)$ is a set of vertices, representing *job types*,
- each vertex $v \in V(\tau)$ is labeled with an ordered tuple $\langle e(v), d(v), \mu(v) \rangle \in (\mathbb{N}_{\geqslant 0} \times \mathbb{N}_{\geqslant 0} \times M(T))$, representing worst-case execution time, relative deadline and mode of the corresponding job type, respectively,
- $E_{\text{cf}}(\tau)$ is a set of directed edges representing task control flow, such that $\mu(u) = \mu(v)$ for each $(u, v) \in E_{\text{cf}}(\tau)$,
- each edge $(u, v) \in E_{\text{cf}}(\tau)$ is labeled with a minimum inter-release separation time $p(u, v) \in \mathbb{N}_{\geqslant 0}$,
- $E_{\text{ms}}(\tau)$ is a set of directed edges representing possible mode switches, such that $\mu(u) \neq \mu(v)$ for each $(u, v) \in E_{\text{ms}}(\tau)$.

We assume that each task $\tau \in T$ satisfies the *frame separation property*, a generalization of the *constrained deadlines* concept for sporadic tasks. In other words, for each vertex $u \in V(\tau)$ and $(u, v) \in E_{\text{cf}}(\tau)$ we have $d(u) \leqslant p(u, v)$.

Note that, by the above definition, $E_{\text{cf}}(\tau)$ and $E_{\text{ms}}(\tau)$ are disjoint sets, that $(V(\tau), E_{\text{cf}}(\tau))$ is a directed graph with up to k disjoint subgraphs (one subgraph per mode of the task), and that $(V(\tau), E_{\text{ms}}(\tau))$ is a directed multipartite graph (colorable with one color per mode).

B. Semantics

An MS-DRT system consists of a number of tasks that all run in the same mode at any particular time point, i.e., the modes are system wide. While running in any mode, an MS-DRT task τ behaves in the same way as an ordinary DRT task: the task runs by traversing the graph $(V(\tau), E_{\text{cf}}(\tau))$, releasing a job every time a vertex is visited. More formally, a job is defined by a triple $(r, e, d) \in \mathbb{R}^3$, representing the job's release time, execution-time budget and absolute deadline, respectively. A valid job sequence $[(r_1, e_1, d_1), (r_2, e_2, d_2), \ldots]$ is generated by τ if there is a path $(\pi_1, \pi_2, \ldots)$ through $(V(\tau), E_{\text{cf}}(\tau))$ such that for all i

- $e_i = e(\pi_i)$,
- $d_i = r_i + d(\pi_i)$,
- $r_{i+1} \geqslant r_i + p(\pi_i, \pi_{i+1})$.

As edges in $E_{\text{cf}}(\tau)$ only go between vertices labeled with job types of the same mode, traversing $(V(\tau), E_{\text{cf}}(\tau))$ never causes the task to switch mode.

For any point in time and any mode μ, if the latest visited vertex u of each task $\tau \in T$ has an outgoing mode-switch edge $(u, v) \in E_{\text{ms}}(\tau)$ with $\mu(v) = \mu$, then an event can trigger the system to switch to mode μ. At that time point, each task τ immediately and synchronously switches to the new mode μ through one of the edges in $E_{\text{ms}}(\tau)$, chosen non-deterministically if there are several such edges. Note that the model does not specify the origin of the events triggering mode switches, but rather just says that such events can arrive at any time. Any event-triggering scheme chosen by the system designer is then valid for the model. For example, mode-switch events can be emitted due to the run-time behavior of the tasks themselves, or due to execution-time overruns of jobs. They could also be the result of errors or faults, or come from external sources.

The mode switch protocol is of a (generalized) mixed-criticality style, meaning that if the last released job of τ is still active (released, but not finished), it immediately has its parameters changed to that of the job type at the target vertex. In this way, the job types labeled on any two vertices u, v, for which $(u, v) \in E_{\text{ms}}(\tau)$, can be thought of as representing different versions of *the same job*. Note that when a mode-switch edge $(u, v) \in E_{\text{ms}}(\tau)$ is taken, no new job is released at vertex v. The job type labeled on v serves to update the parameters of a job from u that is still active as follows.

1) The job's total execution-time budget is changed from $e(u)$ to $e(v)$, but is not replenished. If the job has already executed for at least $e(v)$ time units, it is immediately considered to be finished.
2) The job's absolute deadline is changed to be $d(v)$ time units after its release time.

Jobs that are active during a mode switch are called *carry-over jobs*. Note that a job is still eligible to become a carry-over job at the time point where its execution-time budget reaches zero; this allows modeling of mode switches due to execution-time overruns.

After the switch, the tasks can go on to generate new job sequences by continuing to traverse $(V(\tau), E_{\text{cf}}(\tau))$, now only being able to visit vertices labeled with job types of the new mode. The inter-release separation constraints hold across mode switches. In other words, if the last released job of τ (active or not) was released at time t in a previous mode, then the first control-flow edge $(v, w) \in E_{\text{cf}}(\tau)$ to be followed in the new mode can be taken earliest at time $t + p(v, w)$.

A system may start with any mode as the initial one, and with any vertices with job types of that mode as the initial vertices of the tasks.[1]

III. Examples

In this section we present some example tasks, showing a few of the properties that can be modeled with the MS-DRT task model. In the figures we draw the control-flow edges as straight arrows and the mode-switch edges as wiggly arrows. The colors help reading, but carry no semantic information.

[1]In practice, systems will likely have just one or a few well-defined initial states. However, allowing the system to start in any reachable state does not negatively affect schedulability. As the goal of the model is to over-approximate all the possible behaviors of the system, we found it counterproductive to add syntax to needlessly restrict the behaviors of the model.

Example III.1 (Dual-criticality tasks)**.** *Figure 1 shows some tasks that are similar to ordinary mixed-criticality tasks [2], but with some additional semantics that can not be expressed in the original model. Upon an execution-time overrun, the system would switch from mode* LO *to mode* HI.

- τ_1 *is equivalent to a dual-criticality sporadic task that gets its execution-time budget increased at a switch to the high-criticality mode (*HI*).*
- τ_2 *will instead drop any active job at a mode switch, and after a delay start a less intensive sporadic workload. Recall that the inter-release separation constraints hold transparently across mode switches, so the extra dummy job at u_3 is introduced to ensure that u_4 is visited no earlier than 100 time units after the mode switch as opposed to 100 time units after the last job release at u_1.*
- τ_3 *will stop releasing new jobs after a mode switch, but must finish any active job that it has at that time; the time given to finish the last job is increased to 70 time units instead of the 30 time units that are normally given.*
- τ_4 *is a direct extension of a two-vertex DRT task to the dual-criticality semantics with different execution-time estimates.*
- τ_5 *represents overhead from the mode switch (e.g., reordering priority queues) by creating a one-off job that must be executed immediately after the switch. Note that the "worst-case" behavior of τ_5 is to defer the release of a job at z_1 until the time point where a switch to* HI *occurs, and then immediately transforming it with the parameters at z_2. It is often the case that the model will have many possible behaviors that are irrelevant for the concrete system, as long as the behaviors of the system are a subset of those of the model it is not an issue for performing safety analysis.*

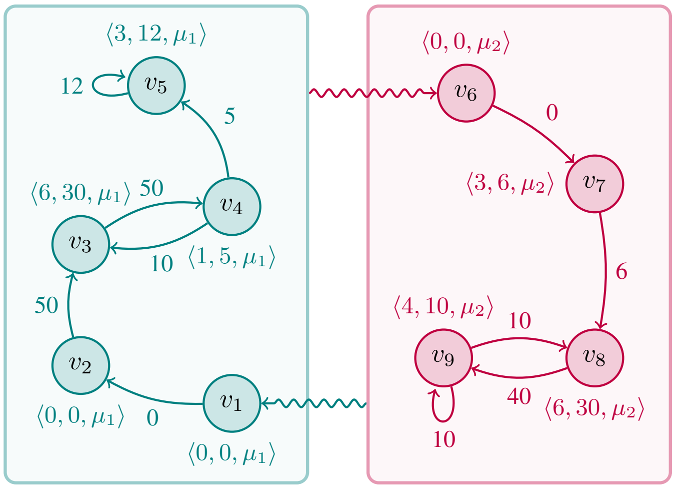


Fig. 2. An example task with coarse-grained mode switching.

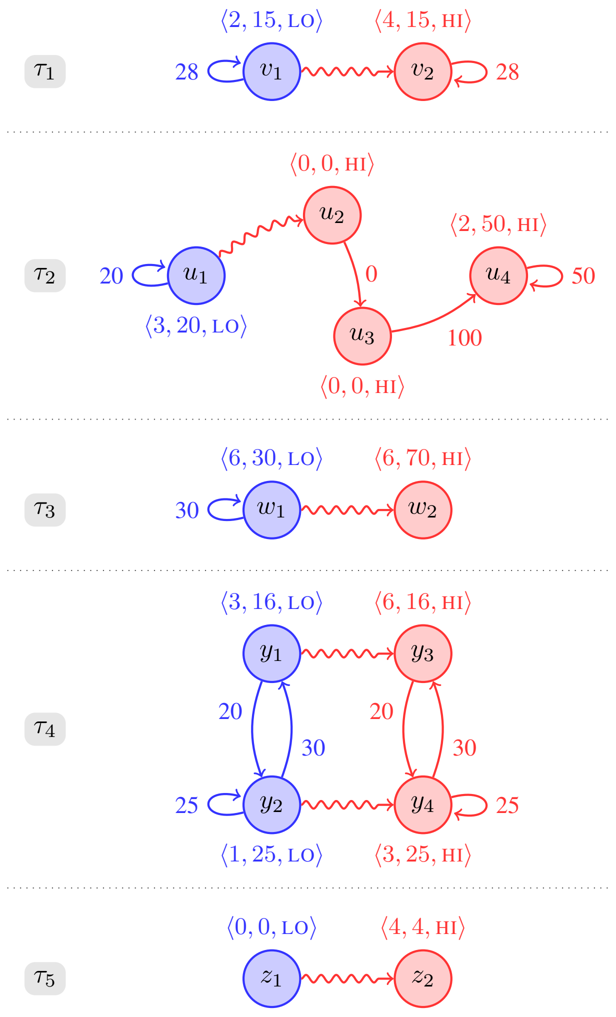


Fig. 1. Some example tasks of the ordinary dual-criticality style.

Example III.2 (Coarse-grained mode switching)**.** *More traditional, coarse-grained mode switching can also be modeled in MS-DRT. Figure 2 shows a task that only has a single entry point per mode. The rectangular boxes are syntactic sugar expressing that the outgoing edges have copies from each of the vertices inside the box. From any of the vertices $v_1, \ldots, v_5$, the task can switch to mode μ_2 by going to v_6. Any active job is dropped at v_6, and some initial work has to be performed immediately at v_7. The task can switch to v_1 in mode μ_1 from any of the vertices $v_6, \ldots, v_9$, in the process dropping any active job and delaying at least 50 time units before continuing to release the first non-dummy job at v_3. Note that while the model dictates that all active jobs of the task, if any, are dropped at a mode switch, a perfectly valid behavior of the system that is modeled would be to only switch to a new mode when the task actually has no active jobs.*

Example III.3 (Period-adapting tasks)**.** *In this example we model two tasks, shown in Figure 3, that periodically read some sensor values and release jobs to process the readings. Depending on the values that are read, the resulting jobs have* *different execution-time requirements. Some readings result in "small jobs" with an execution-time requirement of 1 time unit, and other readings result in "big jobs" that take up to 4 time units to complete.*

The tasks are together adapting to the current sensor values by switching to different modes where the periodicity of the readings are matched against the execution-time requirements of the resulting jobs. In mode μ_1, both tasks are reading and processing small jobs at the highest possible pace in v_1 and u_1, respectively. If τ_1 reads a sensor value implying a big job, it goes to the dummy job v_5 and triggers a mode switch to mode μ_2, upon which the dummy job is transformed into a big job at v_2. At the same time, τ_2 must also switch to μ_2 and goes to u_3, where the current (small) job continues with the same execution-time budget. The period of the sensor readings in μ_2 is changed to 5 in order to match the total execution-time requirements of the jobs. If τ_1 later reads a sensor value implying a small job, it will go to v_9 and trigger a mode switch back to μ_1. Task τ_2 will switch back to μ_1 through u_8, allowing it to finish its current job with the deadline given to it before reverting back to the smaller period of sensor readings. Similarly, the system switches to mode μ_3 if τ_2 reads sensor values resulting in big jobs, and to μ_4 if both tasks do.

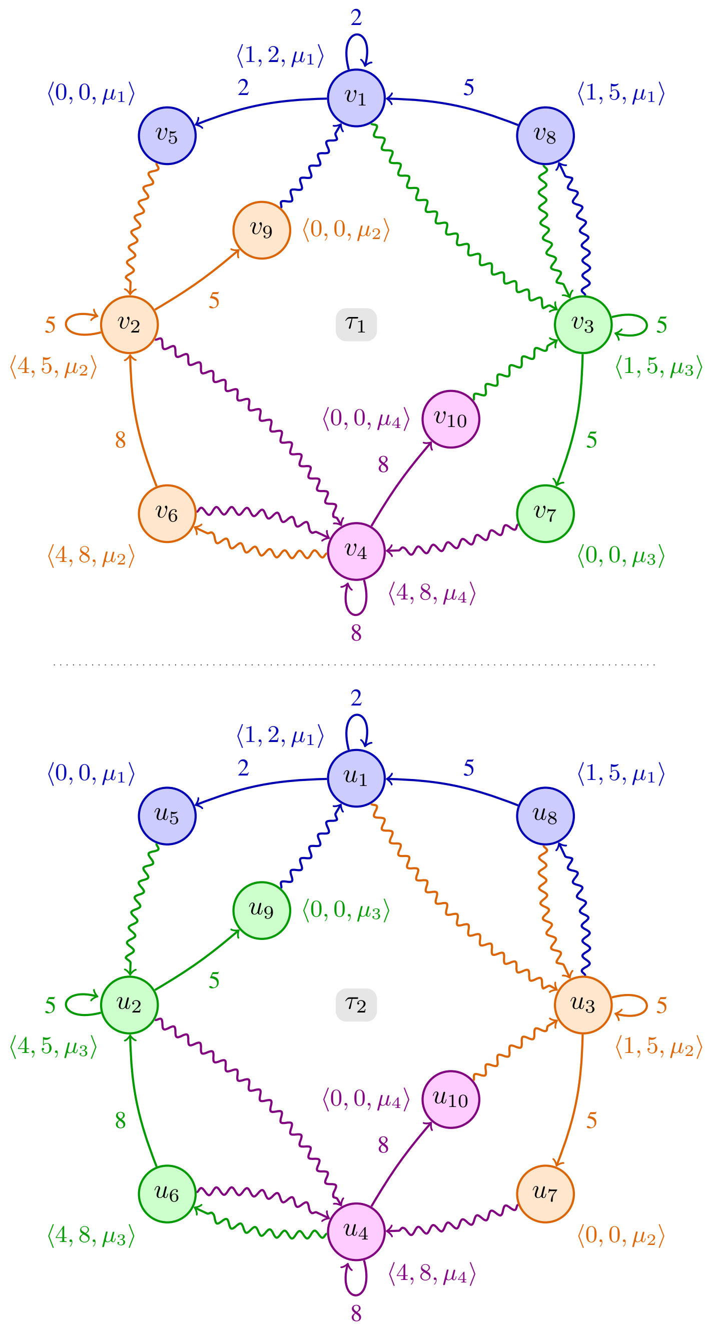
IV. Analysis

In this section we will briefly outline an EDF schedulability analysis of MS-DRT task systems on uniprocessors. It is based on ideas from previously published EDF schedulability analyses of regular DRT task systems [1] and mixed-criticality sporadic task systems [6], [4].

Following the analysis for the generalized mixed-criticality sporadic task model [4], we define the *mode structure* $G(T)$ of an MS-DRT task system T as the directed graph (V, E) where $V = M(T)$ is the set of modes and E contains edges for the possible mode switches.[2] That is, $(\mu_i, \mu_j) \in E$ if and only if each task $\tau \in T$ has vertices u, v such that $(u, v) \in E_{\text{ms}}(\tau)$ and $\mu(u) = \mu_i$ and $\mu(v) = \mu_j$. For example, Figure 4 shows the mode structure for the tasks in Example III.3.

To reduce the complexity of the schedulability analysis we analyze each mode and mode switch separately. For each mode μ_j we will analyze its schedulability during all possible time intervals that do *not* include a mode switch. For every mode switch $(\mu_i, \mu_j) \in E$ we will analyze the schedulability of μ_j for all possible time intervals that start with a switch from μ_i to μ_j, over-approximating any workload that can be carried over from the previous mode.

The analysis is based on finding *demand bound functions* for each task. The demand bound functions must safely over-approximate the total execution demand of all jobs from the task that together can have their entire scheduling windows (from release time to deadline) within a time interval of a given length. Let $\text{dbf}_{\mu_j}(\tau, \ell)$ denote a demand bound function for τ in mode μ_j for any time interval of length ℓ that do not contain



v_1
$\langle 1, 2, \mu_1 \rangle$
2
v_5
$\langle 0, 0, \mu_1 \rangle$
2
v_8
$\langle 1, 5, \mu_1 \rangle$
5
v_9
$\langle 0, 0, \mu_2 \rangle$
v_2
$\langle 4, 5, \mu_2 \rangle$
5
5
τ_1
v_3
$\langle 1, 5, \mu_3 \rangle$
5
v_{10}
$\langle 0, 0, \mu_4 \rangle$
5
8
8
v_6
$\langle 4, 8, \mu_2 \rangle$
v_4
$\langle 4, 8, \mu_4 \rangle$
8
v_7
$\langle 0, 0, \mu_3 \rangle$
u_1
$\langle 1, 2, \mu_1 \rangle$
2
u_5
$\langle 0, 0, \mu_1 \rangle$
2
u_8
$\langle 1, 5, \mu_1 \rangle$
5
u_9
$\langle 0, 0, \mu_3 \rangle$
u_2
$\langle 4, 5, \mu_3 \rangle$
5
5
τ_2
u_3
$\langle 1, 5, \mu_2 \rangle$
5
u_{10}
$\langle 0, 0, \mu_4 \rangle$
5
8
8
u_6
$\langle 4, 8, \mu_3 \rangle$
u_4
$\langle 4, 8, \mu_4 \rangle$
8
u_7
$\langle 0, 0, \mu_2 \rangle$

Fig. 3. Two tasks that dynamically adapt to each other's requirements.

a mode switch. Let $\text{dbf}_{\mu_i \to \mu_j}(\tau, \ell)$ instead denote a demand bound function for task τ in mode μ_j for any time interval of length ℓ that start with a switch from μ_i to μ_j. Only the latter kind of demand bound function will be over-approximate. The pessimism there comes from assuming the worst-case behavior in the previous mode when bounding the demand of carry-over jobs; this is the price that we pay for analyzing the modes in relative separation. Experimental results from [4], where a similar source of pessimism exists, shows that the resulting analysis procedure still offers good performance compared to

[2]In [4] the mode structure is constrained to be a directed acyclic graph, here it can be any directed graph.

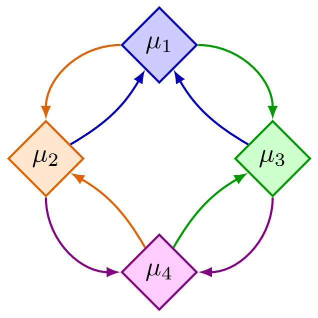


Fig. 4. The mode structure of the tasks in Example III.3.

other methods. Considering modes in separation also enables us to handle systems with cyclic mode switches with relative ease, as we only need to look to the immediately previous mode to over-approximate the demand of carry-over jobs.

The demand bound functions described above can be used in a schedulability test as expressed by the following proposition.

Proposition IV.1. *An MS-DRT task system T with mode structure $G(T) = (V, E)$ is schedulable by EDF if the following two conditions hold for all $\mu_j \in V$:*

$$\forall \ell \geqslant 0 : \sum_{\tau \in T} \mathrm{dbf}_{\mu_j}(\tau, \ell) \leqslant \ell, \tag{1}$$

$$\forall \mu_i \in \mathrm{pred}(\mu_j), \forall \ell \geqslant 0 : \sum_{\tau \in T} \mathrm{dbf}_{\mu_i \to \mu_j}(\tau, \ell) \leqslant \ell, \tag{2}$$

where $\mathrm{pred}(\mu_j) \overset{\text{def}}{=} \{\mu_i \mid (\mu_i, \mu_j) \in E\}$.

Conceptually, the above proposition checks the schedulability of each mode in complete isolation. It can be thought of as $|M(T)|$ separate schedulability tests. For each mode μ_j, we ensure in condition (1) that the demand of workload released entirely inside μ_j is schedulable and in condition (2) that workload including carry-over jobs from all possible predecessor modes is schedulable. Considering this, a proof of the above proposition can be made very similar to, e.g., the proof of Theorem 1 in [11] and is omitted for space reasons.

A. Computing demand bound functions

The computation of demand bound functions for MS-DRT tasks is based on the method for computing such functions for regular DRT tasks [1]. A quick review of this method is presented below.

1) Demand bound function computation for DRT: For any path $\pi = (\pi_1, \ldots, \pi_m)$ through a DRT graph, the execution-time demand $e(\pi)$ and deadline $d(\pi)$ of π is defined as

$$e(\pi) \overset{\text{def}}{=} \sum_{i=1}^{m} e(\pi_i),$$

$$d(\pi) \overset{\text{def}}{=} \sum_{i=1}^{m-1} p(\pi_i, \pi_{i+1}) + d(\pi_m).$$

To compute a (precise) demand bound function of a DRT task for a time interval length ℓ, we must find the maximum $e(\pi)$ for all paths π through the graph with $d(\pi) \leqslant \ell$.

As the number of paths through a graph grows exponentially with the path length, a path abstraction called *demand tuples* was introduced in [1] to alleviate this problem. Each demand tuple can abstractly represent several concrete paths that are equivalent for the purposes of computing the demand bound function. The most basic demand tuple abstraction of a path $\pi = (\pi_1, \ldots, \pi_m)$ is simply a triple $\langle e(\pi), d(\pi), \pi_m \rangle$ with the execution demand, deadline and final vertex of the path. The same demand tuple would abstract all paths that share these three properties. The demand tuples can be used instead of concrete paths for traversing the task graph by extending them with more vertices. If (u, v) is an edge of the graph and $\langle e, d, u \rangle$ a demand tuple for path π, then $\langle e', d', v \rangle$ with $e' = e + e(v)$ and $d' = d - d(u) + p(u, v) + d(v)$ is a demand tuple for π extended by v.

The problem of finding the maximum $e(\pi)$ for any path π with $d(\pi) \leqslant \ell$ can then be transformed into finding $\max\{e \mid \langle e, d, v \rangle$ demand tuple with $d \leqslant \ell\}$. To generate all demand tuples required to compute the demand bound function for all relevant values of the interval length ℓ, we first find an upper bound $\ell_{\max}$ on the values of ℓ that must be considered for schedulability. We then create demand tuples $\langle e(v), d(v), v \rangle$ for all vertices (or 0-length paths) v, and then iteratively extend each demand tuple $\langle e, d, v \rangle$ with more vertices as explained above, as long as $d \leqslant \ell_{\max}$. Duplicate demand tuples can be discarded on the fly, and other optimizations can be applied as well [1]. It was shown in [1] that a pseudo-polynomial $\ell_{\max}$ can be found assuming that the utilization of the task set is bounded by a constant strictly smaller than 1. There are therefore at most pseudo-polynomially many demand tuples to consider, and the process of computing the demand bound function is of pseudo-polynomial time complexity.

2) Demand bound function computation for MS-DRT: The above method for computing demand bound functions can be used for finding the intra-mode demand bound functions $\mathrm{dbf}_{\mu_j}(\tau, \ell)$ for MS-DRT tasks τ. The initial demand tuples $\langle e(v), d(v), v \rangle$ are created from vertices $v \in V(\tau)$ with $\mu(v) = \mu_j$, and the tuples are extended using edges from $E_{\mathrm{cf}}(\tau)$. In fact, one can apply the demand bound function computation for DRT tasks directly on the subgraph $(V(\mu_j, \tau), E_{\mathrm{cf}}(\tau))$, where $V(\mu_j, \tau) \overset{\text{def}}{=} \{v \in V(\tau) \mid \mu(v) = \mu_j\}$. Computing a bound $\ell_{\max}$ on the values of ℓ that must be considered for mode μ_j in Proposition IV.1 can be done exactly as in [1], using the set $\{(V(\mu_j, \tau), E_{\mathrm{cf}}(\tau)) \mid \tau \in T\}$ as the set of DRT tasks.

It is more difficult to compute the demand bound functions $\mathrm{dbf}_{\mu_i \to \mu_j}(\tau, \ell)$ for time intervals starting with a mode switch, as these must account for possible carry-over jobs from the previous mode. To achieve this, we adapt the ideas from [6], [4] for bounding the demand of carry-over jobs for mixed-criticality sporadic tasks. Consider a task τ that switches from mode μ_i to μ_j through an edge $(u, v) \in E_{\mathrm{ms}}(\tau)$, as illustrated in Figure 5.

We will assume that mode μ_i is schedulable when computing $\mathrm{dbf}_{\mu_i \to \mu_j}(\tau, \ell)$, i.e., that all deadlines are met in μ_i. By

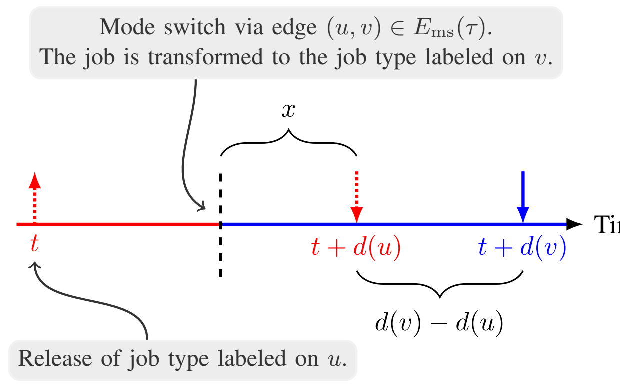

Mode switch via edge $(u, v) \in E_{\text{ms}}(\tau)$.
The job is transformed to the job type labeled on v.
x
Time
t
$t + d(u)$
$t + d(v)$
$d(v) - d(u)$
Release of job type labeled on u.

Fig. 5. Task τ switches from μ_i to μ_j trough an edge $(u, v) \in E_{\text{ms}}(\tau)$. The parameters of the active job are updated with the parameters of the job type labeled on vertex v. The job's remaining execution-time budget in μ_i is at most x time units, assuming that μ_i is schedulable.

reasoning similar to that in [4], this is a safe assumption for the purposes of finding demand bound functions to be used for schedulability analysis as in Proposition IV.1. With this assumption, we can conclude that if the mode switch happens x time units before the carry-over job's absolute deadline in μ_i, the job can have at most x time units left of its execution-time budget in μ_i at that time point. The job's remaining execution time budget as it enters mode μ_j is then at most $e(x, u, v) \overset{\text{def}}{=} \min(x, e(u)) + e(v) - e(u)$ time units. The length of the time interval until its new absolute deadline in μ_j is $d(x, u, v) \overset{\text{def}}{=} x + d(v) - d(u)$ time units. For the purposes of computing the demand bound function, we consider the carry-over job in μ_j as a new job released at the time of the mode switch, with execution-time budget and deadline as above. Such a job, if it exists, must be the first job to contribute demand to a time interval starting at a mode switch. A straightforward approach to computing $\text{dbf}_{\mu_i \to \mu_j}(\tau, \ell)$ is then to create the initial demand tuples

$$\langle e(x, u, v), d(x, u, v), v\rangle$$

for all $(u, v) \in E_{\text{ms}}(\tau)$ and $x \in [0, e(u)]$ such that $\mu(u) = \mu_i$ and $\mu(v) = \mu_j$. Note, however, that $x = e(u)$ corresponds to the "worst-case" demand in the sense that a smaller x leads to $e(x, u, v)$ decreasing by the same amount as $d(x, u, v)$. If the demand of the entire task set can be too high for some interval length ℓ_1 with $x \leqslant e(u)$, then it can also be too high for another interval length $\ell_2 \geqslant \ell_1$ when $x = e(u)$.[3] It is therefore enough for our purposes to create the above demand tuples with $x = e(u)$.

If the task has no active job at the time of the switch from μ_i to μ_j, then the first job to contribute demand in μ_j must be of a job type labeled on a vertex w such that $(v, w) \in E_{\text{cf}}(\tau)$ and $(u, v) \in E_{\text{ms}}(\tau)$ for some u, v with $\mu(u) = \mu_i$ and $\mu(v) = \mu_j$. For all such vertices w, we need to create demand tuples $\langle e(w), d(w), w\rangle$. These demand tuples, together with the ones for the carry-over jobs above, are all that are needed for safely abstracting the demand of all 0-length paths that start at a switch from μ_i to μ_j. These initial demand tuples can then be extended with additional vertices in the same manner as for regular DRT tasks, and ultimately be used to construct a $\text{dbf}_{\mu_i \to \mu_j}(\tau, \ell)$ for use in Proposition IV.1.

[3]This reasoning only holds under the assumption of a unit-speed dedicated processor as in Proposition IV.1. If we instead use some supply bound function (that is piecewise-linear between integer points) as a model of the computing platform, we might have to create demand tuples for all integers $x \in \{0, \dots, e(u)\}$.

V. Conclusions

We have presented MS-DRT, a task model that supports the modeling of complex arrival and synchronization patterns with state-based mode changes. The mode switching protocol is of a mixed-criticality style, implying that MS-DRT generalizes both previous graph-based and mixed-criticality (sporadic) task models. MS-DRT also enables the modeling of many types of systems that fall outside of what is usually considered for mixed-criticality scheduling, some examples of such systems were shown in Section III. We have outlined how EDF schedulability analysis for MS-DRT can be performed by combining ideas from previous methods that use demand bound functions. We believe that MS-DRT offers a type of expressiveness not seen in commonly used models of mode-switching systems. At the same time, it offers the possibility of schedulability analysis that is significantly more efficient than for powerful timing models such as timed automata [12].

The schedulability analysis for mixed-criticality sporadic tasks that we adapted for MS-DRT can be greatly improved by a parameter-tuning preprocessing procedure [6], [4]. This tuning artificially decreases some of the relative deadlines in order to shift demand between the demand bound functions of different modes. A similar procedure can be applied to MS-DRT tasks, although the process of tuning is considerably more involved. As future work we plan to tackle this challenge, as well as to work out the remainder of the schedulability analysis in more technical detail.

References

[1] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *RTAS*, 2011, pp. 71–80.
[2] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *RTSS*, 2007, pp. 239–243.
[3] S. Baruah, "Certification-cognizant scheduling of tasks with pessimistic frequency specification," in *SIES*, 2012, pp. 31–38.
[4] P. Ekberg and W. Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems," *Real-Time Systems*, 2013.
[5] D. Harel, "Statecharts: a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231 – 274, 1987.
[6] P. Ekberg and W. Yi, "Bounding and shaping the demand of mixed-criticality sporadic tasks," in *ECRTS*, 2012, pp. 135–144.
[7] S. Baruah, K., V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, "Scheduling real-time mixed-criticality jobs," *IEEE Transactions on Computers*, Jul. 2011.
[8] H. Li and S. Baruah, "An algorithm for scheduling certifiable mixed-criticality sporadic task systems," in *RTSS*, 2010, pp. 183 –192.
[9] S. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "Mixed-criticality scheduling of sporadic task systems," in *ESA*, 2011, pp. 555–566.
[10] S. Baruah, A. Burns, and R. Davis, "Response-time analysis for mixed criticality systems," in *RTSS*, 2011, pp. 34 –43.
[11] S. Baruah, D. Chen, S. Gorinsky, and A. Mok, "Generalized multiframe tasks," *Real-Time Systems*, vol. 17, pp. 5–22, 1999.
[12] E. Fersman, P. Krcal, P. Pettersson, and W. Yi, "Task automata: Schedulability, decidability and undecidability," *Information and Computation*, vol. 205, no. 8, pp. 1149 – 1172, 2007.