EDF Schedulability Analysis for an Extended Timing Definition Language
T. Kloda, B. d’Augsbourg, L. Santinelli

To cite this version:
T. Kloda, B. d’Augsbourg, L. Santinelli. EDF Schedulability Analysis for an Extended Timing Definition Language. 9th IEEE International Symposium on Industrial Embedded Systems, Jun 2014, PISA, Italy. hal-01067909

HAL Id: hal-01067909
https://hal-onera.archives-ouvertes.fr/hal-01067909
Submitted on 24 Sep 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
EDF Schedulability Analysis for an Extended Timing Definition Language

Tomasz Kloda, Bruno d’Ausbourg, Luca Santinelli
ONERA Toulouse, name.surname@onera.fr

Abstract—In a time-triggered system, activities like task releasing, operational mode switches, sensor readings and actuations are all initiated at predetermined time instants. This paper proposes an extension of the TDL (Timing Definition Language) time-triggered compositional framework, and presents, based on the widely-applied methods, a condition for its schedulability. The schedulability condition developed accounts for multiple concurrently executing modules, multiple operational modes and mode switches. This way the system schedulability can be guaranteed in any execution condition.

I. INTRODUCTION

The development of embedded software is a highly platform dependent process. The main difficulty lies in both formulating the functional specification of the system and correctly determining its temporal behavior. While the former is facilitated by high-level programming languages which abstract from many hardware aspects, getting the expected temporal characteristic of the system involves usually much more efforts due to the implementation of scheduling policies, synchronization, and inter-processes communication protocols.

To efficiently manage these two crucial aspects for system correctness, time-triggered languages are devised for embedded programming. These languages clearly separate the functional part of applications and their timing definition. Applications are specified through two descriptions: their timing definition, expressed in a time-triggered language, and the functional code of tasks, expressed in any programming language. At the final stage, a dedicated compiler generates, based on both descriptions, a ready-to-run executable for a selected target-platform. This allows system designers and developers to focus on the functional aspects instead of the platform (hardware and operating system) without being interested in where, how and when tasks are actually scheduled: on platforms with a single or many CPUs, on platforms with a preemptive priority scheduling or not.

Time-triggered languages provide a programming abstraction which was firstly introduced within the Giotto programming language [13]. Giotto assigns to each task a Logical Execution Time (LET) [14] which defines the precise instants when the task exchanges data with the other tasks and with its environment. A task is invoked and reads its input ports at the beginning of its LET interval, then performs a computation whose results are made available on its output ports exactly at the end of the LET. Figure 1 shows the difference between the logical and physical execution of a task. The observable temporal behavior of a task is independent from the platform related factors such as processing speed, scheduling policy and communication protocols.

Fig. 1: Logical Execution Time and its physical execution.

Timing Definition Language (TDL) [9], [10], [26], a successor of Giotto, extends these concepts by allowing a decomposition of large real-time systems into concurrently executing tasks sets named modules. Each TDL module runs in one operational mode at a time and can switch the modes independently of other modules.

A. Contributions

While enabling timing and value predictability of programs, Giotto and TDL can be further extended towards a more flexible and more realistic application modeling. In both frameworks, the task LET and period are equal. Introducing a task model with an initial offset and the LET of a task terminating before the end of its period, is what we call Extended Timing Definition Language (E-TDL). This new E-TDL framework is presented in Section II.

With E-TDL it is necessary to guarantee safe execution during multimodal operation inside every module. In section I-B are cited some methods [27], [11] that can be adapted to our case in order to provide a schedulability condition. They address mainly event-triggered systems where the execution of an activity is triggered by the occurrence of an event whose arrival cannot be predefined beforehand. Hence, it can be supposed that in all modules the events which produce the biggest demand may arrive at the same time.

Time-triggered systems observe the state of the controlled object only at specified time instants and initiate appropriate activities only at these instants. Time instants of tasks activations are precisely defined and it is known whether tasks are launched simultaneously or not. Consequently it may be avoided to consider the synchronous case in analysis when it is well known that this case never occurs.

Compositional analysis of such systems should be able to exhibit the relations between the start of the intervals in different modules and compare only these that can actually
start at the same time. Therefore, the analyses that are well-suited for event-triggered systems can be overestimating in the time-triggered context.

With this work we propose a sufficient condition for the feasibility of an E-TDL system running on a single CPU under the Earliest Deadline First (EDF) scheduling policy. [15]. In Section IV we describe an offset analysis to identify feasible execution configurations. In Section V we detail the schedulability analysis which makes use of these configurations and applies to time-triggered E-TDL systems.

B. Related Work

Many different protocols and methodologies attempting to ensure the schedulability of a system across mode switches have been proposed in the literature. Protocols known as synchronous [28], [2], [22] do not release new tasks until all old mode tasks are completed. On the contrary, asynchronous protocols, both for Fixed Priority [29], [19], [22] and EDF [1], [7], define that during transition phase the last activations of old mode tasks and new mode tasks can be executed simultaneously. In TDL and E-TDL, systems can be composed of many modules and each module can undergo mode transitions that are defined only in its local scope. For systems designed in a compositional manner, an approach based on the real-time calculus (RTC) [27] is developed in the work of Stoimenov et al. [25]. Fisher [11] proposed a schedulability test for EDF where the allocation of the processing resources for each subsystem is represented by an explicit-deadline periodic resource model [8] and a sporadic task model [17] is chosen. Servers can also dynamically adapt their parameters of resource reservation. The feasibility under multi-moded resource reservation was studied by Santinelli et al. in [23].

Concerning the schedulability analysis of time-triggered systems the most significant contributions were made notably by Farcaș [9] for TDL and Martinek in the Giotto in Ada [16] framework. The latter extends a Giotto’s task model with the notion of deadline but, since Giotto is not a compositional framework, these results cannot be applied in this instance.

II. EXTENDED TDL MODEL

An E-TDL periodic task \( \tau_i = (\Phi_i, C_i, LET_i, T_i) \) is characterized by an offset \( \Phi_i \), a worst-case execution time \( C_i \), a Logical Execution Time \( LET_i \) (that gives its relative deadline \( D_i \)) and a period \( T_i \).

The offset \( \Phi_i \) is restricted to be smaller than the period \( 0 \leq \Phi_i < T_i \) and \( LET_i \) has to be large enough that task could be executed within it, \( C_i \leq LET_i \leq T_i - \Phi_i \). Moreover, the LET semantics does not allow the \( j \)-th instance \( (j \in \mathbb{N}_+) \) of task \( \tau_i \) started in the time interval \((j - 1)T_i, jT_i] \) to be finished after time \( jT_i \). Taking into account the above mentioned constraints, an E-TDL task \( \tau_i \) may be implemented by a periodic real-time task whose absolute releases and deadlines are expressed respectively as \( r_{i,j} = \Phi_i + (j - 1)T_i \leq jT_i \) and \( d_{i,j} = \Phi_i + LET_i + (j - 1)T_i \leq jT_i \).

![Fig. 2: E-TDL task model.](image)

The LET semantics imposes that a task reads periodically its inputs exactly when it is released and delivers its outputs exactly at the end of its LET interval.

An E-TDL system consists of multiple modules running on the same node. All modules of the system, referred hereafter as Modules, run concurrently sharing common processing resources. Each module \( M_i \in \text{Modules} \), from a scheduling point of view, is considered independent from the others.

The basic functional units of time-triggered languages are tasks that periodically execute some piece of code. Several concurrent tasks may be grouped into a mode. Tasks are invoked within the mode at declared frequencies and can be removed or added when switching from one mode to another. They communicate between them as well as with sensors and actuators by means of ports. A mode is invoked and performs appropriate actions whenever the environment is in some specific condition that should be handled by this particular operational mode [18]. With a mode \( m_k \), the set of tasks belonging to \( m_k \) is \( \tau[m_k] \).

At a time module \( M_j \) executes one of its modes from the set \( \text{Modes}[M_j] \). One of them is the initial mode \( m_{init} = \text{Init}[M_j] \) and is executed at the system beginning. Each mode \( m_k \) executes periodically within its mode period \( T[m_k] \). A mode period is restrained to be a common multiple of the periods of all tasks belonging to that mode, \( T[m_k] \equiv n H[m_k] \) with \( n \in \mathbb{N}_+ \) and \( H[m_k] \equiv \text{lcm}\{T_i \mid \tau_i \in \tau[m_k]\} \).

The relative amount of time spent inside a mode \( m_k \) is named mode time \( \delta_k \) and is set to 0 every time a mode period begins. A module state can be described by the tuple \((m_k, \delta_k)\). As mode time \( \delta_k \) evolves within the mode period \( T[m_k] \), \( 0 \leq \delta_k \leq T[m_k] \), appropriate actions like releasing or terminating tasks, are triggered.

Whenever a condition change is detected in the environment or inside the system, the current operating mode stops its activities and a new mode starts instantaneously with a mode set to zero. These transitions are described in E-TDL by mode switches. During execution of mode \( m_k \) the conditions to switch from mode \( m_k \) to mode \( m_{k+1} \) are checked periodically every mode switch period \( T_{sw}(m_k, m_{k+1}) \). The instants at which this check occurs are named mode switch instants, \( \delta_{ms} \). If the conditions are satisfied, the mode switch can take place. Every mode switch period \( T_{sw}(m_k, m_{k+1}) \) is restricted to be a common multiple of all the task periods of \( m_k \):

\[
T_{sw}(m_k, m_{k+1}) \mod H[m_k] = 0 \land \\
\exists n \in \mathbb{N}_+, n T_{sw}(m_k, m_{k+1}) = T[m_k].
\]

Such a choice of mode switch instants entails that a mode switch may occur only when no current mode task is running. This way the new mode can be safely activated without any
delay. In future works we will enhance the mode switch within E-TDL to let it safely happen before \( T_{sw}(m_k, m_{k+1}) \).

Within an actual mode \( m_k \), the modes that may be started at \( \delta_k \) belong to the set \( \text{starting\_modes}(m_k, \delta_k) \) where

\[
\text{starting\_modes}(m_k, \delta_k) = \{ m_{k+1} \mid 3 \text{E-TDL}(m_k, m_{k+1}) : \delta_k \mod T_{sw}(m_k, m_{k+1}) = 0 \} \cup \{ m_k \mid \delta_k = T[m_k] \}.
\]

\[\quad (2)\]

III. COMPOSITIONAL SCHEDULABILITY ANALYSIS FOR E-TDL

The presented E-TDL multi-mode model imposes on each module of the system a set of possible execution patterns that can be observed during the system temporal evolution. In what follows, the behavior of every module is characterized in time and the processing resources the module demands over time are quantified. This permits to provide a sufficient condition for the schedulability of the whole system under mode switches.

A. Execution Trace of an E-TDL Module

In [9] Farcas proposed a mode-switch graph which allows to exhibit all the modes of a given module and the transitions it can undergo. Modes are represented by vertices and mode switches by directed edges in the graph. The mode switch period labels the edge. More formally, the mode switch graph of a module \( M_j \in \text{Modules} \) describes the set of switches that this module can undergo. Each mode switch is defined as a tuple \((m_k, m_{k+1}, T_{sw}(m_k, m_{k+1}))\) where \( m_k, m_{k+1} \in \text{Modes}[M_j] \) and \( T_{sw}(m_k, m_{k+1}) \) is the period of the \((m_k, m_{k+1})\) mode switch. Restarting the same mode after the end of its period can be also considered as a mode switch \((m_k, m_{k+1}, T[m_k])\) and is so depicted on the graph.

The mode switch graph can be defined as the set:

\[
\{ (m_k, m_{k+1}, T_{sw}(m_k, m_{k+1})) \mid m_k, m_{k+1} \in \text{Modes}[M_j], \exists \delta_k : m_{k+1} \in \text{starting\_modes}(m_k, \delta_k) \} \quad (3)
\]

Fig. 3: Example of a mode-switch graph.

During its execution, a module can remain in the current mode or switch to one of its possible successive modes. The behavior of a module can be described by a walk that travels through its mode-switch graph, visiting modes at multiples of mode switch periods, jumping to another modes whenever there is a need for changing mode.

**Definition 1 (Walk in a Mode-Switch Graph)**. A walk \( w = ((m_1, \mu_1), \ldots, (m_n, \mu_n)) \) of an E-TDL mode-switch graph is a sequence of \( n \) pairs where \( m_1, \ldots, m_n \) are the subsequent modes in the mode-switch graph, and \( \mu_1, \ldots, \mu_k \) are the numbers of:
- times the mode switch condition to \( m_{k+1} \) has been checked in \( m_k \) (number of \( T_{sw}(m_k, m_{k+1}) \) spent in \( m_k \)), for \( k < n; \mu_k \in \mathbb{N}_+ \),
- full periods \( T[m_k] \) spent in \( m_k \), for \( k = n; \mu_k \in \mathbb{N}_0 \).

The first mode \( m_1 \) in the walk \( w \) is designated as \( \text{head}(w) \) and the last mode \( m_n \) as \( \text{tail}(w) \). The length of a walk \( w = ((m_1, \mu_1), \ldots, (m_n, \mu_n)) \) can be expressed as:

\[
|w| = \sum_{k=1}^{n} \mu_k T_{sw}(m_k, m_{k+1})
\]

We use \( T_{sw}(m_k, m_{k+1}) \) interchangeably with \( T[m_k] \) for \( k = n \) as there is no next mode in the walk \( w \).

Figure 4 illustrates how an execution trace in the interval \([t_s, t_f]\) can be decomposed. This trace encapsulates a walk \( w \) between two time intervals. Both of these time intervals cover an execution pattern within a single mode run. The interval that precedes \( w \) starts at time \( t_s \) and ends when the module enters mode \( \text{head}(w) \). The interval at the tail of \( w \) starts in mode \( \text{tail}(w) \) at mode time 0 and ends at \( t_f \).

**Definition 2 (Module Execution Trace)**. An execution trace of an E-TDL module \( M_j \in \text{Modules} \) over a time interval \([t_s, t_f]\) is denoted by \( \sigma_{M_j} = (m_s, \delta_s, \delta'_s, w, m_f, \delta_f) \) where
- \( m_s \in \text{Modes}[M_j] \) is the mode executing before walk \( w \),
- \( \delta_s : 0 < \delta_s \leq T[m_s] \) is the mode time related to \( t_s \),
- \( \delta'_s : \delta_s \leq \delta'_s \leq T[m_s] \) is the mode time of mode \( m_s \),
- \( w \) is a walk in the mode-switch graph of module \( M_j \),
- \( m_f \in \text{Modes}[M_j] \) is the mode executing after walk \( w \),
- \( \delta_f : 0 \leq \delta_f < T[m_f] \) is the mode time related to \( t_f \).

It has to be fulfilled that:
- \( \text{head}(w) \in \text{starting\_modes}(m_s, \delta'_s) \),
- if \( \text{tail}(w) \neq m_f \), then \( m_f \in \text{starting\_modes}(\text{tail}(w), T[\text{tail}(w)]) \) and\( \text{tail}(w) \) is executed at least one time at the end of \( w \),
- \( t_f - t_s = \delta'_s - \delta_s + |w| + \delta_f \).

Short intervals, within which no more than one mode period is executed, can be represented as \( \sigma_{M_j} = (m_s, \delta_s, \delta'_s, 0, 0, 0) \).

![Fig. 4: Example of a module execution trace](image-url)

The length of an execution trace \( \sigma_{M_f} \) is given by \( |\sigma_{M_f}| = \delta'_s - \delta_s + |w| + \delta_f \), and mode as well as mode time in which it starts are defined as \( \text{start}(\sigma_{M_j}) = (m_s, \delta_s) \).
When a module $M_j$ runs a walk, it crosses modes. $M_j$ can stay in any mode $m_k$ before starting $m_{k+1}$ for any time that is a multiple of the mode switch period $T_{sw}(m_k, m_{k+1})$. The length of a walk can be expressed as a linear combination of the modes switch periods it spends in each of its modes.

**Definition 3 (Path in a Mode-Switch Graph).** A path $p$ in an E-TDL mode-switch graph is the set of mode switch periods that can be traversed by some walk $w$ that executes tail$(w)$ at least once. Walk $w = ((m_1, \mu_1), \ldots, (m_k, \mu_k), (m_{k+1}, \mu_{k+1}), \ldots, (m_n, \mu_n))$ where $\mu_n \geq 1$, follows a path $p$ if:

$$p = \{T_{sw}(m_k, m_{k+1}) | k < n\} \cup \{T[m_k] | k = n\}.$$  

**B. Resource demand of an E-TDL Module**

To determine the schedulability of an E-TDL system, the resource demand in each module should be precisely quantified.

An E-TDL execution trace is the combination of two types of intervals. Before and after its walk no more than one mode period is executed. The walk itself constitutes a sequence of intervals that span over some number of mode switch periods. Based on the concept of demand function and demand bound function [4], what follows evaluates the resource demands in such intervals with respect to the E-TDL model we are considering. Let $\delta_k$ and $\delta'_k$ be the mode times of the mode $m_k$ such that $0 \leq \delta_k < \delta'_k \leq T[m_k]$. The demand function $df(m_k, \delta_k, \delta'_k)$ within a mode $m_k$ is the cumulative execution time required by all the task instances in $m_k$ having its releases and deadlines between $\delta_k$ and $\delta'_k$.

$$df(m_k, \delta_k, \delta'_k) \equiv \sum_{\tau_i \in \tau[m_k]} \left( \left\lfloor \frac{\delta'_k - \Phi_i - LET_i}{T_i} \right\rfloor - \left\lfloor \frac{\delta_k - \Phi_i}{T_i} \right\rfloor + 1 \right) C_i$$  

For a walk $w$, its demand function can be calculated as:

$$df(w) \equiv \sum_{(m_k, \mu_k) \in w} \mu_k T_{sw}(m_k, m_{k+1}) U(m_k)$$  

where $U(m_k)$ is the processor utilization factor of mode $m_k$:

$$U(m_k) \equiv \sum_{\tau_i \in \tau[m_k]} \frac{C_i}{T_i}.$$  

The foregoing definitions can be applied to express the resource demanded by an E-TDL module execution trace $\sigma_{M_j}$.

**Definition 4 (Demand Function).** For a given E-TDL execution trace $\sigma_{M_j} = (m_s, \delta_s, \delta'_s, w, m_f, \delta_f)$, its demand function $df(\sigma_{M_j})$ is the cumulative processing time required by $M_j$ to execute $\sigma_{M_j}$.

$$df(\sigma_{M_j}) \equiv df(m_s, \delta_s, \delta'_s) + df(w) + df(m_f, 0, \delta_f)$$  

Execution traces depend on the system evolution. During an interval $\Delta$ that starts when module $M_j$ is running mode $m_s$ at mode time $\delta_s$, different execution traces may be observed: each of them is depending on the performed mode switches. Each of these execution traces is characterized by some value of demand function. Only the highest demand function value of these traces can be considered in the further analysis.

**Definition 5 (Maximal Demand Function).** For module $M_j \in \text{Modules}$ maximal demand function $\maxdf_{M_j}(m_s, \delta_s, \Delta)$ denotes the largest value of demand function that characterizes some E-TDL execution trace of length $\Delta$ that starts from mode time $\delta_s$ in mode $m_s \in \text{Modes}[M_j]$.

$$\maxdf_{M_j}(m_s, \delta_s, \Delta) \equiv \max_{\sigma_{M_j}} \{df(\sigma_{M_j}) | \text{start}(\sigma_{M_j}) = (m_s, \delta_s), \|\sigma_{M_j}\| = \Delta\}$$  

The following definition estimates the highest demand function value for an E-TDL execution trace which can start at an arbitrary mode time in any one of the declared modes. We call it maximal demand bound function to recall the notion of demand bound function from [4] and differentiate with respect to it due to the different task model from E-TDL.

**Definition 6 (Maximal Demand Bound Function).** The maximal demand bound function $\mdbf_{M_j}(\Delta)$ denotes the maximal cumulative requirement of computation for module $M_j \in \text{Modules}$ during any time interval $\Delta$ over E-TDL execution traces.

$$\mdbf_{M_j}(\Delta) \equiv \max_{(m_s, \delta_s)} \{\maxdf_{M_j}(m_s, \delta_s, \Delta)\}$$  

**C. Schedulability of an E-TDL System**

We are now able to formulate our first schedulability condition for an E-TDL system under EDF. This condition is derived from the processor demand criterion [4], [6]. The schedulability of a task set where deadlines are less than periods is guaranteed whenever the cumulative demand of the computation made by its tasks in any interval is never larger than the length of this interval. The following theorem adapts this reasoning to the E-TDL framework by cumulating the processor demands over modules. It checks for any time interval if the total amount of processing time requested by all the individual modules to complete their execution traces does not exceed the interval length. Multi-mode E-TDL with mode switches at $T_{sw}$ are included into the following schedulability theorem.

**Theorem 1 (Schedulability of an E-TDL System).** Let be an E-TDL system defined by a set of modules $\text{Modules}$ such that:

$$\sum_{M_j \in \text{Modules}} \max_{m \in \text{Modes}[M_j]} \{U(m)\} \leq 1$$  

If for every time interval $\Delta > 0$ :

$$\sum_{M_j \in \text{Modules}} \mdbf_{M_j}(\Delta) \leq \Delta$$

then the E-TDL system is schedulable under EDF on one processor.

Proof: The proof of this theorem can be derived from Lemma 3.4 in [4]. The authors show that a complete task
system is feasible on one processor if for any time instants \( t_1 \) and \( t_2 \) such that \( 0 \leq t_1 < t_2 \), the processor resources requested in the interval \([t_1, t_2] \) do not exceed the available processing resources in this interval. Since we assume that one unit of computation is available per one unit of time, the total amount of processing time available in this interval is equal to \( t_2 - t_1 \). According to Definition 6, the computational time requested by the tasks of module \( M_j \in \text{Modules} \) in the interval \([t_1, t_2] \) is not larger than \( \text{mdbf}_{M_j}(t_2 - t_1) \). All modules from the set \( \text{Modules} \) compete for the processor time in the interval \([t_1, t_2] \) and therefore the demands from all these modules must be summed up and compared to the available resources.

The sufficient condition from Theorem 1 relies on the sum of maximal demands in all modules. It assumes that these maximal demands are synchronous as in the worst case. In reality, the fact that mode periods, task periods and task offsets are precisely fixed in time within the time-triggered paradigm, can imply that this synchronous worst case does not correspond to any actual activation pattern. In that case the condition overestimates the actual processor demand over modules. The following example shows the schedulability overestimation for time-triggered systems.

**Example 1.** Given an E-TDL system composed of modules \( M_1 \) and \( M_2 \) and two positive-integers \( \pi \) and \( \varepsilon \) such that \( \varepsilon < \pi \). Module \( M_1 \) can run in one of two modes \( \text{Mode}_1(M_1) = \{m_1, m_1'\} \) both having the same mode period: \( T[m_1] = T[m_1'] = \pi \) at the end of which each one can jump to the other mode. Module \( M_2 \) has only one mode \( \text{Mode}_2(M_2) = \{m_2\} \) with the mode period \( T[m_2] = 2\pi \). All the modes are single-task and their task sets are as follows: \( \tau[m_1] = \{\tau_1 = (0, \varepsilon, \pi, \pi), \tau[m_1'] = \{\tau_1' = (\varepsilon, \pi - \varepsilon, \pi - \varepsilon, \pi)\}, \tau[m_2] = \{\tau_2 = (0, 2(\pi - \varepsilon), 2\pi, 2\pi)\}.\]

![Fig. 5: A sample run of the E-TDL system from Example 1.](image)

It can be seen from the above figure that the system is schedulable. However, for the interval \( \Delta = 2\pi \) the condition of Theorem 1, Equation (12), is not satisfied. The demand bound functions over an interval of this length for the individual modules are:

- \( \text{mdbf}_{M_1}(\Delta) = \varepsilon + \pi \) for \( \sigma_{M_1} = (m_1', \varepsilon, \pi, (m_1, 1), m_1, \varepsilon) \)
- \( \text{mdbf}_{M_2}(\Delta) = 2(\pi - \varepsilon) \) for \( \sigma_{M_2} = (m_2, 0, 2\pi, 0, 0, 0) \)

The cumulative demand is \( \text{mdbf}_{M_1}(\Delta) + \text{mdbf}_{M_2}(\Delta) = 3\pi - \varepsilon > 2\pi \) as \( 0 < \varepsilon < \pi \). The summed up demands are associated with intervals that will never completely overlap due to the timing control of tasks in the time-triggered paradigm. Since these demands will never be simultaneous, adding them overestimates the cumulative demand.

In the next sections we make use of the bricks presented so far to study offsets between the traces from distinct modules and develop more accurate schedulability conditions able to better cope with multi-module and multi-mode E-TDL. To reduce the pessimism of Theorem 1 and to provide an exact characterization of a feasible E-TDL system, it is necessary to define properly how execution traces in different modules overlap. It is necessary to study precisely the possible activation patterns of modes that are running concurrently in different modules.

**IV. E-TDL Multi-Module and Multi-Mode Offset Analysis**

**A. Module start time instants**

Let \( (m_{dst}, \delta_{dst}) \) an instantaneous state of a module \( M_j \). This state occurs \( \delta_{dst} \) time after the beginning of \( m_{dst} \). The set \( \text{start}(m_{dst}) \) of absolute time instants at which module \( M_j \) can start mode \( m_{dst} \) is given by the lengths of all the walks that \( M_j \) may travel from its initialization up to the beginning of \( m_{dst} \).

\[
\text{start}(m_{dst}) = \{|w| \mid \text{head}(w) = \text{Init}[M_j], \text{tail}(w) = m_{dst}\}
\]

In the following we give some conditions to exploit tasks, modes, and modules relationships. The schedulability analysis of multi-module multi-mode E-TDL systems will benefit from these conditions. The statements of these conditions are based on paths.

**Definition 7 (Greatest Common Divisor of Path).** The greatest common divisor \( \gcd(p) \) of a path \( p \) is the greatest common divisor of all mode switch periods making this path and of the period of its last mode.

Defining \( T'_{sw}(m_k, m_{k+1}) = T_{sw}(m_k, m_{k+1}) / \gcd(p) \), the length of a walk in Equation (4) can be rewritten as:

\[
|w| = \gcd(p) \sum_{j=1}^{n} \mu_k T'_{sw}(m_k, m_{k+1}).
\]

All the pairs \( T'_{sw}(m_k, m_{k+1}) \) within a path \( p \) are mutually prime, and a walk \( w \) of \( p \) is a linear combination of these coprime integers. From the Frobenius problem statement [21], [24] it follows that, given a set of relatively prime positive integers \( a_1, \ldots, a_n \), any natural number larger than some bound noted as \( F(a_1, \ldots, a_n) \) can be expressed as a linear combination of \( a_1, \ldots, a_n \). Then, for a given path \( p \) the lengths of its potential walks, and so the start instants of the last mode in these walks, fall in the set of all natural multiples of \( \gcd(p) \). A generic destination mode \( m_{dst} \) can begin at the time instants in \( \text{start}(m_{dst}) \), and it is

\[
\text{start}(m_{dst}) \subset \bigcup_{p \in P_{dst}} \{n \gcd(p) \mid n \in \mathbb{N}\}
\]
where $P_{dst}$ is the set of all the paths leading from the initial mode to the destination mode $m_{dst}$. Then $(m_{dst}, \delta_{dst})$ can be observed at time $t = t_{start}(m_{dst}) + \delta_{dst}$ where $t_{start}(m_{dst}) \in \text{start}(m_{dst})$.

B. Time distance between mode starts in distinct modules

We now evaluate the time distances that can separate start instants of the modes running in distinct modules. The proposed approach is derived from the work of Pellizzoni and Lipari [20] that is applied to the analysis of the real-time periodic tasks with offsets.

The following lemma characterizes the time distances between activations of two modes in two distinct modules.

Lemma 1 (Interval between two modes). The time distance between start instants of mode $m_a \in \text{Modes}[M_a]$ and mode $m_b \in \text{Modes}[M_b]$ with $M_a, M_b \in \text{Modules}$ and $M_a \neq M_b$, lies within the set

$$ \bigcup_{p_a \in P_a, p_b \in P_b} \{ \alpha \gcd(p_a \cup p_b) \mid \alpha \in \mathbb{Z} \}. $$

$P_a$ is a set of paths leading from $\text{Init}[M_a]$ to $m_a$ and $P_b$ is a set of paths leading from $\text{Init}[M_b]$ to $m_b$.

Proof: The difference $\Delta_{\text{start}}(m_a, m_b)$ between start instants of mode $m_a$ and mode $m_b$ which were reached by traversing respectively path $p_a$ and path $p_b$ may be expressed using Equation (14):

$$ \Delta_{\text{start}}(m_a, m_b) = t_{start}(m_a) - t_{start}(m_b) = n_1 \gcd(p_a) - n_2 \gcd(p_b) $$

According to the Bézout’s identity [3], for any non zero integers $c$ and $d$ there exist integers $x, y \in \mathbb{Z}$ such that $cx + dy = \gcd(c, d)$ and every integer of the form $cx + dy$ is a multiple of $\gcd(c, d)$. This implies that the values of $\Delta_{\text{start}}(m_a, m_b)$ are multiples of $\gcd(p_a)\gcd(p_b)$. Because the greatest common divisor operation satisfies the property of associativity [12], the last term can be rewritten as $\gcd(p_a \cup p_b)$.

The following lemma refines Lemma 1 by evaluating precisely the time intervals that can separate the start instants of two concurrent modes instances.

Lemma 2 (Interval between two concurrent modes). Given mode $m_a \in \text{Modes}[M_a]$ and mode $m_b \in \text{Modes}[M_b]$ where $M_a, M_b \in \text{Modules}$ such that $M_a \neq M_b$. If the current instance of mode $m_b$ started no later than the current instance of mode $m_a$, then the time distances between starts of these two instances lie inside the following set:

$$ \bigcup_{p_a \in P_a, p_b \in P_b} \{ \alpha \gcd(p_a \cup p_b) \}, $$

where $\alpha \in \mathbb{N} : 0 \leq \alpha \gcd(p_a \cup p_b) < T[m_b]$

Proof: For $\alpha = 0$ the most recent instances of $m_a$ and $m_b$ start exactly at the same time. The previous possible mode’s $m_b$ activation points are moved back by multiples of $\gcd(p_a \cup p_b)$ as explained in Lemma 1. From some $\alpha$ value, start instants of mode $m_b$ belong to the previous $m_b$ instances (terminated before the activation of the current one). Since the only considered instances of mode $m_b$ should be these which run concurrently with $m_a$, the value of $\alpha \gcd(p_a \cup p_b)$ cannot exceed the length of the period $T[m_b]$.

The above relation holds for a pair of modes. We generalize this result to the case of a compound system of more than two modes.

Lemma 3 (Intervals between n modes). Let $\text{Modules}' \subseteq \text{Modules}$ be a set of $n - 1$ E-TDL modules $M_1, \ldots, M_{n-1}$ where $n : 3 \leq n \leq |\text{Modules}|$ and $m_1, \ldots, m_{n-1}$ be $n - 1$ modes each belonging to a different module in $\text{Modules}'$. Let $M_n \in \{ \text{Modules} - \text{Modules}' \}$ and $m_n \in \text{Modes}[M_n]$. Suppose that the current instances of modes $m_1, \ldots, m_{n-1}$ and $m_n$ are running at the same time and every mode $m_k$ ($k \in \mathbb{N} : 1 \leq k \leq n$) has been reached by taking a path $p_k$. Suppose also that mode $m_1$ started its most recent activation no sooner than all the other modes ($\forall k > 1 : \delta_1 \leq \delta_k$).

If each mode $m_k \in \text{Modules}'$ ($1 < k < n$) started at $\alpha_k \gcd(p_1 \cup p_k)$ time before mode $m_1$, then the possible offsets that separate the start of $m_n$ and the start of $m_1$ are given by $\alpha_n \gcd(p_1 \cup p_n)$, $\alpha_n \in \mathbb{N}$ and must satisfy the following conditions:

1. $0 \leq \alpha_n \gcd(p_1 \cup p_n) < T[m_n]$
2. $\forall k \in \mathbb{N} : 1 < k < n, \exists r \in \mathbb{Z} : \alpha_n \gcd(p_1 \cup p_n) = \alpha_k \gcd(p_1 \cup p_k) + r \gcd(p_k \cup p_n)$.

Proof: The first condition results from Lemma 2. The second condition anchors the start instant of $m_n$ with respect to the start instants of the other modes. Each mode $m_k : 1 < k < n$ starts $\alpha_k \gcd(p_1 \cup p_k)$ time units before the start of $m_1$ and is not finished by this time. From Lemma 1 it follows that the start instant of $m_n$ can occur only at time points that lie in the distance $r \gcd(p_k \cup p_n)$ from the start of $m_k$. This relation is expressed by the second condition.

For an E-TDL system defined by a set $\text{Modules}$, the point of interest is to identify all the relations between concurrently running modes $m_1, \ldots, m_{|\text{Modules}|}$ in distinct modules of the system. These modes are reached by walking along paths $p_1, \ldots, p_{|\text{Modules}|}$.

Algorithm 1 VerifyStartInstances($p_1, \ldots, p_n, \alpha_2, \ldots, \alpha_n$)

Require: $\forall k \leq n : p_k$ is a path to mode $m_k$

Require: Lemma 3 holds for $p_1, \ldots, p_{n-1}$ and $\alpha_2, \ldots, \alpha_{n-1}$

Ensure: Lemma 3 holds for $p_1, \ldots, p_n$ and $\alpha_2, \ldots, \alpha_n$

1: if not $0 \leq \alpha_n \gcd(p_1 \cup p_n) < T[m_n]$ or $k_n \notin \mathbb{N}$ then
2: return false
3: end if
4: for $k \leftarrow 2, n - 1$ do
5: $\delta_{\text{diff}} := \alpha_n \gcd(p_1 \cup p_n) - \alpha_k \gcd(p_1 \cup p_k)$
6: $\delta_{\text{rem}} := \delta_{\text{diff}} \mod \gcd(p_k \cup p_n)$
7: if $\delta_{\text{rem}} \neq 0$ then
8: return false
9: end if
10: end for
11: return true
Algorithm 1 makes use of Lemma 3. Given \(\alpha_2, \ldots, \alpha_{n-1}\) that realizes mode starts setup according to Lemma 3 for \(n - 1\) modes, it checks if introduction of \(\alpha_n\) for a \(n\)-th mode characterizes a realizable mode starts setup for these \(n\) modes. We look for a set \(\alpha(\text{Modules}) = (p_1, \ldots, p_{\text{Modules}})\) of all the tuples \((\alpha_2, \alpha_3, \ldots, \alpha_{\text{Modules}})\) meeting conditions of Lemma 3. These tuples represent all the valid modes offsets. The set \(\alpha(\text{Modules})\) can be obtained by applying Algorithm 2, where initially the set \(\alpha(n+2)\) of all \(\alpha_2\) fulfilling Lemma 2 for modes \(m_1\) and \(m_2\) is computed. Then, by incrementing \(n\) from \(n = 3\), the tuples \((\alpha_2, \ldots, \alpha_n)\) are estimated according to Lemma 3, till \(n = |\text{Modules}|\).

Algorithm 2 FindStartInstants\((p_1, \ldots, p_{\text{Modules}})\)

```plaintext
Require: \(\forall M_j \in \text{Modules}: \exists m_k \in \text{Modules}[M_j]\)
Require: \(\forall n \leq |\text{Modules}|: p_n\) is a path to mode \(m_k \in \text{Modules}[M_k]\)
Ensure: \(\forall (\alpha_2, \ldots, \alpha_{\text{Modules}}) \in \alpha(\text{Modules})\) Lemma 3 holds
1: \(\alpha(2) = \{\alpha_2\} \mid \alpha_2\) holds for \(\alpha_2\) and paths \(p_1\) and \(p_2\)
2: for \(n \leftarrow 3, |\text{Modules}|\)
3: \(\alpha(n) = \emptyset\)
4: for all \((\alpha_2, \ldots, \alpha_{n-1}) \in \alpha(n-1)\)
5: for all \(\alpha_n; 0 \leq \alpha_n \gcd(p_1 \cup p_n) < T[m_n]\)
6: if Lemma 3 holds for \((\alpha_2, \ldots, \alpha_{n-1}, \alpha_n)\) then
7: \(\alpha(n) := \alpha(n) \cup (\alpha_2, \ldots, \alpha_{n-1}, \alpha_n)\)
8: end if
9: end for
10: end for
11: end for
12: return \(\alpha(\text{Modules})\)
```

C. Observable Parallel Configurations

An external observer of an E-TDL system can describe its state at a particular time instant of the run as a snapshot. This snapshot describes the modes and mode times in each concurrent module at that moment. We name this snapshot a parallel configuration of the system denoted as \(\zeta = ((m_1, \delta_1), \ldots, (m_n, \delta_n))\) where \(n = |\text{Modules}|, \forall M_j \in \text{Modules} m_k \in \text{Modules}[M_j]\) and \(\delta_k\) is the current mode time in \(m_k\).

A parallel configuration is observable if its modes offsets are in accordance with conditions of Lemma 3 (mode offsets can be found by applying Algorithm 2).

Definition 8 (Observable Parallel Configuration). Given an E-TDL system composed of \(n\) Modules \((n = |\text{Modules}|)\) a parallel configuration \(\zeta = ((m_1, \delta_1), \ldots, (m_n, \delta_n))\) is observable if there exists \((\alpha_2, \ldots, \alpha_N) \in \alpha(n)(p_1, \ldots, p_N)\) such that:

\[\forall k; 1 < k \leq n: \]
\[
(\delta_k - \delta_1 + T[m_1]) \mod T[m_k] = \alpha_k \gcd(p_1 \cup p_k)
\]

Each of these observable parallel configurations will be analyzed and checked for feasibility.

It may happen that the start instant of some mode can occur at any mode time of any other concurrent mode. Given modes \(m_1, \ldots, m_n\) running in parallel such that each mode \(m_k\) \((1 \leq k \leq n)\) has been reached by path \(p_k\), then a mode \(m_x\) \((1 \leq x \leq n)\) can start its execution at whatever mode time of any other mode \(m_k\) if \(\forall k \neq x: \gcd(p_x \cup p_k) = 1\). Furthermore, if the above condition is met not only for some of concurrent modes but for all of them, it can be deduced that any combination of mode times in different modes defines an observable parallel configuration. Given modes \(m_1, \ldots, m_n\) running in parallel such that each mode \(m_k\) can be reached by path \(p_k\) \(\in P_k\) where \(P_k\) is the set of all the paths leading from the initial mode to mode \(m_k\). Any mode \(m_k\) can start its execution at any time instant of concurrent modes if \(\exists P \subseteq \prod_{k=1}^{n} P_k : \forall p_k, p'_k \in P, \gcd(p_k \cup p'_k) = 1\).

In that case all the possible parallel configurations are observable. Consequently, the execution traces with the largest processor demand can be simultaneous. Then Theorem 1 can be used without any overestimation. Otherwise, the following exact analysis can be performed.

V. E-TDL Exact EDF Schedulability

The feasibility condition we derive for an E-TDL system running under EDF on one processor makes use of offsets and task relationships. In order to be schedulable, for any observable parallel configuration, the maximal processing demands of execution traces that begin at that particular configuration cannot exceed the processing resources in any time interval. The following theorem guarantees schedulability of an E-TDL system.

Theorem 2 (Exact Schedulability of an E-TDL System). Let be an E-TDL system defined by a set of \(n\) modules Modules \((M_k \in \text{Modules}, 1 \leq k \leq n)\) such that:

\[
\sum_{k=1}^{n} \max_{m \in \text{Modules}[M_k]} \{U(m)\} \leq 1
\]

If for every observable parallel configuration \(\zeta = ((m_1, \delta_1), \ldots, (m_n, \delta_n))\) and every time interval \(\Delta > 0:\)

\[
\sum_{k=1}^{n} \text{maxdf}_{M_k}(m_k, \delta_k, \Delta) \leq \Delta
\]

where \((m_k, \delta_k)\) is the state of module \(M_k\) in the observable parallel configuration \(\zeta\), then the E-TDL system is schedulable under EDF on one processor.

Proof: The proof of this theorem is similar to the proof of Theorem 1 and can be also derived from Lemma 3.4 in [4]. We designate by \(t_1\) a time instant at which a parallel configuration \(\zeta\) was observed. For \(\Delta > 0\), the maximal demand function \(\text{maxdf}_{M_k}(m_k, \delta_k, \Delta)\) quantifies the greatest possible resources that can be requested by module \(M_k \in \text{Modules}\) in the interval \([t_1, t_2]\) such that \(\Delta = t_2 - t_1\) and \(M_k\) is running in \((m_k, \delta_k)\) at the time instant \(t_1\). The total amount of processing time available in the interval \([t_1, t_2]\) is equal to \(\Delta\). By summing up the maximal demands from the particular modules we obtain the resources that must be granted for E-TDL system in the time interval \([t_1, t_2]\). If these resources do not exceed \(\Delta\), E-TDL system is schedulable.
A. Feasibility Bound

In this section we derive an upper bound for the time interval $\Delta$ in Theorem 2 so that condition 19 becomes fully computationally tractable. If an E-TDL system is not schedulable, then some of its tasks will miss its deadline in the time interval that is no longer than this bound.

Every interval $\Delta$ can be seen as a composition of three adjacent intervals: $\Delta_1, \Delta_2$ and $\Delta_3$. Interval $\Delta_2$ spans over the time from the first to the last hyperperiod point in $\Delta$, whereas $\Delta_1$ and $\Delta_3$ cover the pieces of execution occurring respectively before and after $\Delta_2$. Figure 6 illustrates the above described $\Delta$ decomposition.

Fig. 6: Decomposition of $\Delta$ into three intervals.

The maximum value of demand function in any interval $\Delta_1$ for a module $M_j$, denoted by $df_{M_j}(\Delta_1)$, is given by:

$$\max df_{M_j}(\Delta_1) = \max_{m_k} df_{M_j}(m_k, r_{min}(m_k), H[m_k]) \leq \max_{m_k} (U(m_k)H[m_k])$$

where $r_{min}(m_k)$ is the earliest release of some task from mode $m_k \in Modes[M_j]$ occurring after the beginning of this mode. In like manner, the maximum value of demand function for any interval $\Delta_3$, denoted by $df_{M_j}(\Delta_3)$, can be found:

$$\max df_{M_j}(\Delta_3) = \max_{m_k} df_{M_j}(m_k, 0, d_{max}(m_k)) \leq \max_{m_k} (U(m_k)H[m_k])$$

where $d_{max}(m_k)$ is the latest deadline of some task from mode $m_k \in Modes[M_j]$ before its hyperperiod $H[m_k]$. The maximum value of demand function in any interval $\Delta_2$, covering full executions of the hyperperiods of subsequent modes from the first hyperperiod in the first mode to the start of the last hyperperiod in the last mode inside interval $\Delta$, is denoted by $df_{M_j}(\Delta_2)$ such that:

$$\max df_{M_j}(\Delta_2) \leq \Delta \max_{m_k} U(m_k)$$

where $m_k \in Modes[M_j]$.

As there is no tasks that can be started in one of these intervals and having deadline in another, for any $\Delta$ and any valid mode time $\delta_k$ in mode $m_k$ from module $M_j$ the following relation is derived:

$$\Delta \max_{m_k} U(m_k) + 2 \max_{m_k} (U(m_k)H[m_k]) \geq \max df_{M_j}(\Delta_1) + \max df_{M_j}(\Delta_2) + \max df_{M_j}(\Delta_3) \geq \max df_{M_j}(m_k, \delta_k, \Delta)$$

where $m_k \in Modes[M_j]$.

Similarly to the reasoning proposed by Baruah [4], [5], if the system is not schedulable, there exists some $\Delta$ for which the second condition of Theorem 2 is violated:

$$\sum_{M_j \in Modules} \max df_{M_j}(m_k, \delta_k, \Delta) > \Delta$$

Taking into account beforehand obtained properties bounding function $\max df_{M_j}(m_k, \delta_k, \Delta)$, an overflow observed in $\Delta$ implies that:

$$\sum_{M_j \in Modules} \left( \Delta \max_{m_k} U(m_k) + 2 \max_{m_k} (U(m_k)H[m_k]) \right) > \Delta$$

Thus, it is sufficient to check the feasibility for the intervals that are no longer than that:

$$\Delta < \frac{2 \times \sum_{M_j \in Modules} \max_{m_k} U(m_k)}{1 - \sum_{M_j \in Modules} \max_{m_k} U(m_k)}$$

where $m_k \in Modes[M_j]$.

VI. Test Case

A sample E-TDL system is studied to gain insight into the key aspects of the proposed EDF schedulability analysis.

The test case is a 3-modules E-TDL system, where the modules $M_1, M_2, M_3 \in Modules$ execute on the same node. Each module defines its modes together with modes switches as it is indicated on Figure 7 (the topmost mode in each module is considered to be the initial mode). Periods, mode switch periods as well as all the task parameters making up the particular modes are presented in Table 1.

Fig. 7: A 3-modules E-TDL system and its mode switches

<table>
<thead>
<tr>
<th>Module</th>
<th>mode</th>
<th>$T$</th>
<th>$T_{sw}$</th>
<th>tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>$m_{11}$</td>
<td>10</td>
<td>10</td>
<td>$\tau_{111}(0, 2, 6, 10), \tau_{112}(0, 1, 5, 5)$</td>
</tr>
<tr>
<td>$M_1$</td>
<td>$m_{12}$</td>
<td>8</td>
<td>-</td>
<td>$\tau_{121}(2, 1, 2, 8)$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>$m_{21}$</td>
<td>4</td>
<td>4</td>
<td>$\tau_{211}(0, 1, 4, 4), \tau_{221}(1, 1, 1, 8)$</td>
</tr>
<tr>
<td>$M_3$</td>
<td>$m_{31}$</td>
<td>8</td>
<td>-</td>
<td>$\tau_{311}(2, 1, 1, 8)$</td>
</tr>
</tbody>
</table>

Tab. 1: Modes and tasks properties

On the first attempt, we apply Theorem 1 to verify system schedulability. Since the sum of the maximal utilization factors for the modes in distinct modules is not greater than 1, we verify condition given by Equation (12). In Figure 8, a series of plots illustrates maximal demand bound functions $mdbf_{M_1}$,
mdbf_{M_2}, mdbf_{M_3}, and their sum which is compared to the length of the interval \( \Delta \). As can be seen, the condition of Theorem 1 cannot be satisfied in \( \Delta = 1 \) and \( \Delta = 2 \).

For \( \Delta = 1 \), the resource demands of traces \( \sigma_{M_2} = (m_{22}, 1, 2, 0, 0, 0) \) and \( \sigma_{M_3} = (m_{31}, 2, 2, 0, 0, 0) \) are equal to one. The demand function of every execution trace of length 1 in \( M_1 \) is equal to 0.

For \( \Delta = 2 \), the demand functions of traces \( \sigma_{M_1} = (m_{11}, 2, 4, 0, 0, 0), \sigma_{M_2} = (m_{22}, 1, 3, 0, 0, 0), \sigma_{M_3} = (m_{31}, 2, 4, 0, 0, 0), \sigma_{M_4} = (m_{31}, 1, 3, 0, 0, 0) \) are equal to one.

With the help of the offset analysis and Theorem 2, it can be verified if the revealed above execution traces may start actually at the same time, as assumed by Theorem 1. We verify whether the parallel configurations corresponding to the starts of these execution traces are observable or not.

For \( \Delta = 1 \), the above identified parallel configurations for which the system is not schedulable are given as \( ((m_{11}, 1), (m_{22}, 1), (m_{31}, 2)) \) where \( (m_{11}, 1) \) can be an arbitrary state of module \( M_1 \). Consequently, we have to study the time distances that can separate starts of concurrent instances of modes \( m_{22} \) and \( m_{31} \).

As a first step, for every module, the paths leading to the appropriate mode from the initial mode must be found. Table 2 shows these paths together with their greatest common divisor values (path to \( m_{12} \) is not considered for \( \Delta = 1 \)).

<table>
<thead>
<tr>
<th>Path</th>
<th>( \text{gcd} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{12} )</td>
<td>2</td>
</tr>
<tr>
<td>( m_{11} \rightarrow m_{12} )</td>
<td>2</td>
</tr>
<tr>
<td>( p_{22} )</td>
<td>4</td>
</tr>
<tr>
<td>( m_{31} \rightarrow m_{22} )</td>
<td>4</td>
</tr>
<tr>
<td>( p_{31} )</td>
<td>8</td>
</tr>
</tbody>
</table>

Tab. 2: Greatest common divisor of paths

The greatest common divisors \( \text{gcd}(p_a \cup p_b) \) for pairs of paths leading to modes in different modules are calculated in Table 3. According to Lemma 1, for \( p_{22} = 4 \) and \( p_{31} = 8 \), every time distance separating starts of these two modes is a multiple of \( \text{gcd}(p_{22} \cup p_{31}) = 4 \). From Lemma 2 we obtain

\[
\alpha(p_{22}, p_{31}) = \{0, 1\}. \text{ Mode } m_{31} \text{ can start either at the same time as } m_{22} (0 \times \text{gcd}(p_{22} \cup p_{31})) \text{ or } 4 \text{ time units before (1 \times \text{gcd}(p_{22} \cup p_{31}))). Equation (18) is not satisfied by any coefficient } \alpha(p_{22}, p_{31}) \text{ for module } M_2 \text{ state } (m_{22}, 1) \text{ and module } M_3 \text{ state } (m_{31}, 2), \text{ since } (2 - 1 + 8) \mod 8 \notin \{0, 4\}. \text{ A parallel configuration where these two module states coincide is not observable and therefore the execution traces } \sigma_{M_2} \text{ and } \sigma_{M_3} \text{ cannot start at the same time instant. It can be concluded that the system is schedulable in the interval } \Delta = 1.

For interval \( \Delta = 2 \), the parallel configurations for which the schedulability of the system cannot be guaranteed by Theorem 1 are as follows:

\[
\{(m_{12}, 2), (m_{22}, \delta_{22}), (m_{31}, \delta_{31})\}
\]

\( \delta_{22} \in \{0, 1\}, \delta_{31} \in \{1, 2\} \)

Since all these configurations occur during concurrent execution of modes \( m_{12}, m_{22}, \) and \( m_{31} \), the relationship between their starts should be established.

We apply Lemma 2 to paths leading to modes \( m_{12} \) and \( m_{22} \) (see Table 2) under the assumption that the current instance of mode \( m_{12} \) is invoked no earlier than the current instance of \( m_{22} \). It results in values of \( \alpha_2 \in \alpha^{(2)}(p_{12}, p_{22}) \) which correspond to the distances \( (\alpha_2 \times \text{gcd}(p_{12} \cup p_{22})) \) between starts of \( m_{12} \) and \( m_{22} \) concurrent instances. By considering path leading to mode \( m_{31} \), from Lemma 3, we have \( \alpha^{(3)}(p_{12}, p_{22}, p_{31}) \) of tuples \( (\alpha_2, \alpha_3) \). Each tuple \( (\alpha_2, \alpha_3) \) describes a mode activation pattern in which \( m_{22} \) and \( m_{31} \) started respectively \( \alpha_2 \times \text{gcd}(p_{12} \cup p_{22}) \) and \( \alpha_3 \times \text{gcd}(p_{12} \cup p_{31}) \) instants before \( m_{12} \). Table 4 shows all valid tuples \( (\alpha_2, \alpha_3) \) together with possible walks starting at the initialization of every module and resulting in the activation pattern reflected by this tuple.

<table>
<thead>
<tr>
<th>(( \alpha_2, \alpha_3 ))</th>
<th>( \text{example walk} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (0, 0) )</td>
<td>( (m_{11}, 4) ), ( (m_{21}, 10) ), ( (m_{31}, 5) )</td>
</tr>
<tr>
<td>( (0, 2) )</td>
<td>( (m_{11}, 4), (m_{21}, 4), (m_{22}, 2), (m_{31}, 2) )</td>
</tr>
<tr>
<td>( (1, 1) )</td>
<td>( (m_{11}, 1), (m_{21}, 2), (m_{31}, 1) )</td>
</tr>
<tr>
<td>( (1, 3) )</td>
<td>( (m_{11}, 3), (m_{21}, 5), (m_{22}, 1), (m_{31}, 9) )</td>
</tr>
<tr>
<td>( (0, 2) )</td>
<td>( (m_{11}, 4), (m_{21}, 5), (m_{22}, 2), (m_{31}, 2) )</td>
</tr>
<tr>
<td>( (3, 1) )</td>
<td>( (m_{11}, 4), (m_{21}, 1), (m_{31}, 1) )</td>
</tr>
<tr>
<td>( (0, 0) )</td>
<td>( (m_{11}, 3), (m_{21}, 4), (m_{22}, 1), (m_{31}, 0) )</td>
</tr>
</tbody>
</table>

Tab. 4: Example walks for all \( (\alpha_2, \alpha_3) \in \alpha^{(3)}(p_{12}, p_{22}, p_{31}) \)

Having values of \( \alpha^{(3)}(p_{12}, p_{22}, p_{31}) \) at our disposal, we define all the observable parallel configurations for concurrent execution of modes \( m_{12}, m_{22}, \) and \( m_{31} \). Table 5 represents,
for module’s $M_1$ state $(m_{12}, 2)$, the possible module $M_2$ and $M_3$ states that satisfy Equation (18).

\[
\begin{array}{ccccccccc}
\delta_{22} & 2 & 2 & 4 & 4 & 6 & 6 & 0 & 0 \\
\delta_{31} & 2 & 6 & 4 & 0 & 2 & 6 & 4 & 0 \\
\end{array}
\]

Tab. 5: Parallel configurations $(m_{12}, 2), (m_{22}, \delta_{22}), (m_{12}, \delta_{31})$

It can be seen from Table 5 that the parallel configurations at which the execution traces requesting 1 unit of computation over interval $\Delta = 2$ start are never activated simultaneously. Therefore, the sum of maximal demand functions in $\Delta = 2$, for any observable parallel configuration, is never larger than 2. The system is schedulable according to Theorem 2.

VII. CONCLUSION AND FUTURE WORK

In this paper we extended the TDL task model into E-TDL, and proposed the condition for the feasibility of an E-TDL system scheduled by EDF on one processor. A methodology to exhibit all the possible timing configurations of the system in presence of mode switches has been introduced. Based on the well-founded concept of processor demand we provided techniques to describe resource requirement of such a system.

One of the issues to be addressed in the future is the reduction of the complexity of our solution. This can be achieved by optimizing computation of demand functions and investigating if there exist some point from which they can be periodic. The feasibility bound, in certain cases, could be also estimated more tightly. If in the tested interval the existence of the point where all the modes from distinct modules terminate synchronously can be stated, it would not be necessary to continue the analysis beyond this point.

Another interesting proposal may be running all the modules in temporal isolation. The question that should be answered then would be the choice of the most suitable resource reservation pattern.

Concerning the reactivity of the system, we suggest that mode switches may operate not only at instants that are hyperperiods but also at some possible other idle instants in the mode. Some tasks whose completion is not vital for the consistency of the system, might be aborted across the mode switches.

REFERENCES

torica, 16:143–147, 1996.
[28] K. Tindell, A. Burns, and A. Wellings. Mode changes in priority pre-