

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://onera.hal.science/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.

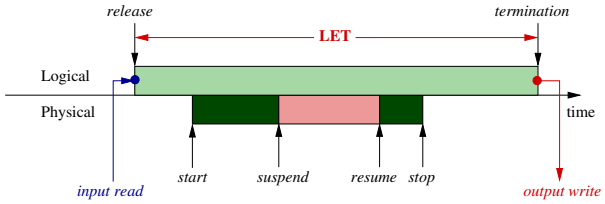


Fig. 1: Logical Execution Time and its physical execution.

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 Farcas [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 Φ_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 Φ_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 τ_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 τ_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 $d_{i,j} = \Phi_i + LET_i + (j-1)T_i \leq jT_i$.

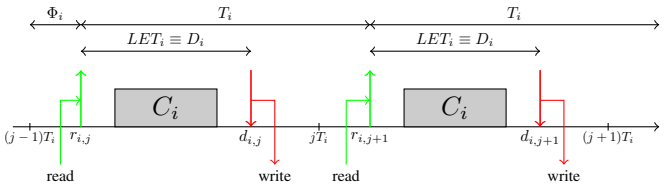


Fig. 2: E-TDL task model.

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 $M_j \in Modules$, 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 $Modes[M_j]$. One of them is the *initial mode* $m_{init} = Init[M_j]$ 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, $T[m_k] \stackrel{\text{def}}{=} n\ H[m_k]$ with $n \in \mathbb{N}_+$, and $H[m_k] \stackrel{\text{def}}{=} lcm\{T_i \mid \tau_i \in \tau[m_k]\}$.

The relative amount of time spent inside a mode m_k is named *mode time* δ_k and is set to 0 every time a mode period begins. A module state can be described by the tuple (m_k, δ_k). As mode time δ_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 time 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*, δ_{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:

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

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 δ_k belong to the set $starting_modes(m_k, \delta_k)$ where

$$starting_modes(m_k, \delta_k) = \{m_{k+1} \mid \exists T_{sw}(m_k, m_{k+1}) : \delta_k \bmod T_{sw}(m_k, m_{k+1}) = 0\} \cup \{m_k \mid \delta_k = T[m_k]\}. \quad (2)$$

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 $M_j \in Modules$ 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 $m_k, m_{k+1} \in Modes[M_j]$ 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 $(m_k, m_k, T[m_k])$ and is so depicted on the graph.
The mode switch graph can be defined as the set:

$$\{(m_k, m_{k+1}, T_{sw}(m_k, m_{k+1})) \,|\, m_k, m_{k+1} \in Modes[M_j], \exists \delta_k : m_{k+1} \in starting_modes(m_k, \delta_k)\} \quad (3)$$

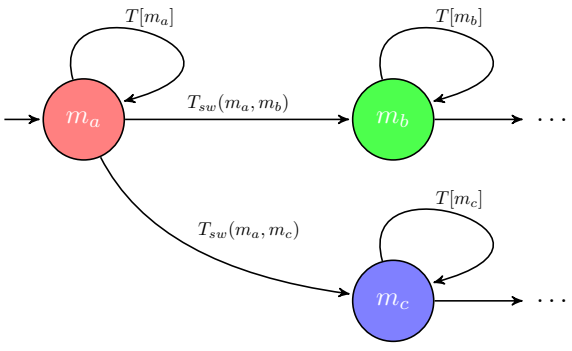


Fig. 3: Example of a mode-switch graph.

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 $head(w)$ and the last mode m_n as $tail(w)$. The length of a walk $w = ((m_1, \mu_1), \ldots, (m_n, \mu_n))$ can be expressed as:

$$|w| = \sum_{k=1}^{n} \mu_k T_{sw}(m_k, m_{k+1}) \quad (4)$$

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 $head(w)$. The interval at the tail of w starts in mode $tail(w)$ at mode time 0 and ends at t_f.

Definition 2 (Module Execution Trace). *An execution trace of an E-TDL module $M_j \in Modules$ 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*

- *$m_s \in Modes[M_j]$ 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,*
- *$m_f \in Modes[M_j]$ 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:

- *$head(w) \in starting_modes(m_s, \delta'_s)$,*
- *$if\ tail(w) \neq m_f$, then $m_f \in starting_modes(tail(w), T[tail(w)])$ and $tail(w)$ 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 $\sigma_{M_j} = (m_s, \delta_s, \delta'_s, \emptyset, \emptyset, 0)$.

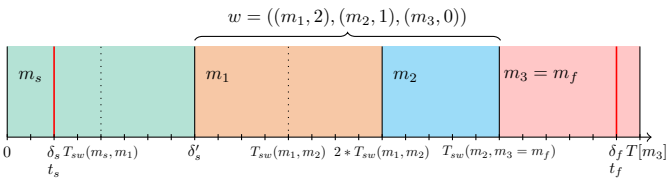


Fig. 4: Example of a module execution trace

The length of an execution trace σ_{M_j} is given by $|\sigma_{M_j}| = \delta'_s - \delta_s + |w| + \delta_f$, and mode as well as mode time in which it starts are defined as $start(\sigma_{M_j}) = (m_s, \delta_s)$.

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 evaluation 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 δ_k and δ'_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 δ_k and δ'_k.

$$df(m_k, \delta_k, \delta'_k) \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau[m_k]} \left(\left\lfloor \frac{\delta'_k - \Phi_i - LET_i}{T_i} \right\rfloor - \left\lceil \frac{\delta_k - \Phi_i}{T_i} \right\rceil + 1 \right)_0 C_i \tag{5}$$

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

$$df(w) \stackrel{\text{def}}{=} \sum_{(m_k, \mu_k) \in w} \mu_k T_{sw}(m_k, m_{k+1}) U(m_k) \tag{6}$$

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

$$U(m_k) \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau[m_k]} \frac{C_i}{T_i}. \tag{7}$$

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

$$df(\sigma_{M_j}) \stackrel{\text{def}}{=} df(m_s, \delta_s, \delta'_s) + df(w) + df(m_f, 0, \delta_f) \tag{8}$$

Execution traces depend on the system evolution. During an interval Δ that starts when module M_j is running mode m_s at mode time δ_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 higest demand function value of these traces can be considered in the further analysis.

Definition 5 (Maximal Demand Function). *For module $M_j \in Modules$ 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 Δ that starts from mode time δ_s in mode $m_s \in Modes[M_j]$.*

$$maxdf_{M_j}(m_s, \delta_s, \Delta) \stackrel{\text{def}}{=} \max_{\sigma_{M_j}} \{df(\sigma_{M_j}) | start(\sigma_{M_j}) = (m_s, \delta_s), |\sigma_{M_j}| = \Delta\} \tag{9}$$

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 $M_j \in Modules$ during any time interval Δ over E-TDL execution traces.*

$$mdbf_{M_j}(\Delta) \stackrel{\text{def}}{=} \max_{(m_s, \delta_s)} \{maxdf_{M_j}(m_s, \delta_s, \Delta)\} \tag{10}$$

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 $Modules$ such that:*

$$\sum_{M_j \in Modules} \max_{m \in Modes[M_j]} \{U(m)\} \leq 1 \tag{11}$$

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

$$\sum_{M_j \in Modules} mdbf_{M_j}(\Delta) \leq \Delta \tag{12}$$

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 $M_j \in Modules$ in the interval $[t_1, t_2]$ is not larger than $mdbf_{M_j}(t_2 - t_1)$. All modules from the set $Modules$ 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 π and ε such that $\varepsilon < \pi$. Module M_1 can run in one of two modes $Modes[M_1] = \{m_1, m_1'\}$ 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 $Modes[M_2] = \{m_2\}$ with the mode period $T[m_2] = 2\pi$. All the modes are single-task and their task sets are as follows: $\tau[m_1] = \{\tau_1 = (0, \varepsilon, \varepsilon, \pi)\}$, $\tau[m_1'] = \{\tau_1' = (\varepsilon, \pi - \varepsilon, \pi - \varepsilon, \pi)\}$, $\tau[m_2] = \{\tau_2 = (0, 2(\pi - \varepsilon), 2\pi, 2\pi)\}$.*

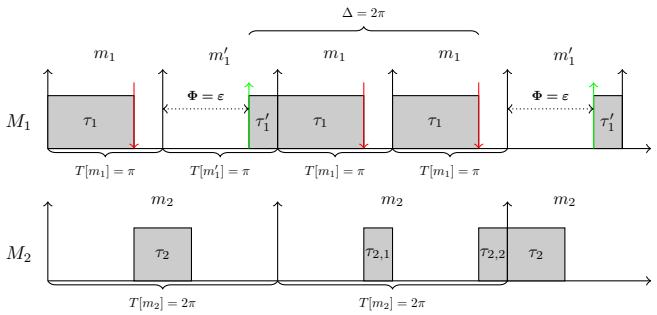


Fig. 5: A sample run of the E-TDL system from Example 1.

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:

- *$mdbf_{M_1}(\Delta) = \varepsilon + \pi$ for $\sigma_{M_1} = (m_1', \varepsilon, \pi, (m_1, 1), m_1, \varepsilon)$*
- *$mdbf_{M_2}(\Delta) = 2(\pi - \varepsilon)$ for $\sigma_{M_2} = (m_2, 0, 2\pi, \emptyset, \emptyset, 0)$*

The cumulative demand is $mdbf_{M_1}(\Delta) + mdbf_{M_2}(\Delta) = 3\pi - \varepsilon > 2\pi$ 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}, δ_{dst}) an instantaneous state of a module M_j. This state occurs δ_{dst} time after the beginning of m_{dst}. The set $start(m_{dst})$ 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}.

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

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

$$|w| = \gcd(p) \sum_{j=1}^{n} \mu_k T'_{sw}(m_k, m_{k+1}). \tag{13}$$

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 $start(m_{dst})$, and it is

$$start(m_{dst}) \subset \bigcup_{p \in P_{dst}} \{n \gcd(p) \mid n \in \mathbb{N}\} \tag{14}$$

where P_{dst} is the set of all the paths leading from the initial mode to the destination mode m_{dst}. Then (m_{dst}, δ_{dst}) can be observed at time $t = t_{start}(m_{dst}) + \delta_{dst}$ where $t_{start}(m_{dst}) \in start(m_{dst})$.

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 $m_a \in Modes[M_a]$ and mode $m_b \in Modes[M_b]$ with $M_a, M_b \in Modules$ and $M_a \neq M_b$, lies within the set*

$$\bigcup_{p_a \in P_a, p_b \in P_b} \{\alpha \gcd(p_a \cup p_b) \mid \alpha \in \mathbb{Z}\}. \tag{15}$$

P_a is a set of paths leading from $Init[M_a]$ to m_a and P_b is a set of paths leading from $Init[M_a]$ to m_b.

Proof: The difference $\Delta_{start}(m_a, m_b)$ 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):

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

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 $\Delta_{start}(m_a, m_b)$ are multiples of $\gcd(\gcd(p_a), \gcd(p_b))$. 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 $m_a \in Modes[M_a]$ and mode $m_b \in Modes[M_b]$ where $M_a, M_b \in Modules$ 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:*

$$\bigcup_{p_a \in P_a, p_b \in P_b} \{\alpha \gcd(p_a \cup p_b)\}, \tag{16}$$

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 α 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 modules.

Lemma 3 (Intervals between n modes). *Let $Modules' \subsetneq Modules$ be a set of $n-1$ E-TDL modules $M_1, \ldots, M_{n-1}$ where $n : 3 \leq n \leq |Modules|$ and $m_1, \ldots, m_{n-1}$ be $n-1$ modes each belonging to a different module in $Modules'$. Let $M_n \in \{Modules - Modules'\}$ and $m_n \in Modes[M_n]$. 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 $m_k \in Modules'$ ($1 < k < n$) started $\alpha_k \gcd(p_1 \cup p_k)$ 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 $Modules$, the point of interest is to identify all the relations between concurrently running modes $m_1, \ldots, m_{|Modules|}$ in distinct modules of the system. These modes are reached by walking along paths $p_1, \ldots, p_{|Modules|}$.

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

Require: $\forall 1 \leq k \leq n : p_k$ 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: $diff := \alpha_n \gcd(p_1 \cup p_n) - \alpha_k \gcd(p_1 \cup p_k)$
6: $rem := diff \bmod \gcd(p_k \cup p_n)$
7: **if** $rem \neq 0$ **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 α_n for a n-th mode characterizes a realizable mode starts setup for these n modes. We look for a set $\alpha^{(|Modules|)}(p_1, \ldots, p_{|Modules|})$ of all the tuples $(\alpha_2, \alpha_3, \ldots, \alpha_{|Modules|})$ meeting conditions of Lemma 3. These tuples represent all the valid modes offsets. The set $\alpha^{(|Modules|)}(p_1, \ldots, p_{|Modules|})$ can be obtained by applying Algorithm 2, where initially the set $\alpha^{(n=2)}$ of all α_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 $n = |Modules|$.

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

Require: $\forall M_j \in Modules : \exists m_k \in Modes[M_j]$
Require: $\forall 1 \leq k \leq |Modules| : p_k$ is a path to mode $m_k \in Modes[M_k]$
Ensure: $\forall(\alpha_2, \ldots, \alpha_{|Modules|}) \in \alpha^{(|Modules|)}$ Lemma 3 holds

```
α^(2) = { α_2 | Lemma 2 holds for α_2 and paths p_1 and p_2 }
for n ← 3, |Modules| do
    α^(n) := ∅
    for all (α_2, ..., α_{n-1}) ∈ α^(n-1) do
        for all α_n : 0 ≤ α_n gcd(p_1 ∪ p_n) < T[m_n] do
            if Lemma 3 holds for (α_2, ..., α_{n-1}, α_n) then
                α^(n) := α^(n) ∪ (α_2, ..., α_{n-1}, α_n)
            end if
        end for
    end for
end for
return α^(|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 $n = |Modules|$, $\forall M_j \in Modules\ m_k \in Modes[M_j]$ and δ_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 $Modules$ ($n = |Modules|$) 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:*

$$\forall k, 1 < k \leq n : \quad (\delta_k - \delta_1 + T[m_1]) \bmod T[m_k] = \alpha_k \gcd(p_1 \cup p_k) \tag{17}$$

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 $\exists P \in \prod_{k=1}^{n} P_k : \forall p_k, p_{k'} \in P, \gcd(p_k \cup p_{k'}) = 1$.

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 $Modules$ ($M_k \in Modules, 1 \leq k \leq n$) such that:*

$$\sum_{k=1}^{n} \max_{m \in Modes[M_k]} \{U(m)\} \leq 1 \tag{18}$$

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

$$\sum_{k=1}^{n} maxdf_{M_k}(m_k, \delta_k, \Delta) \leq \Delta \tag{19}$$

where (m_k, δ_k) is the state of module M_k in the observable parallel configuration ζ, 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 ζ was observed. For $\Delta > 0$, the maximal demand function $maxdf_{M_k}(m_k, \delta_k, \Delta)$ quantifies the greatest possible resources that can be requested by module $M_k \in Modules$ in the interval $[t_1, t_2]$ such that $\Delta = t_2 - t_1$ and M_k is running in (m_k, δ_k) at the time instant t_1. The total amount of processing time available in the interval $[t_1, t_2]$ is equal to Δ. 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 Δ, E-TDL system is schedulable. ■

A. Feasibility Bound

In this section we derive an upper bound for the time interval Δ in Theorem 2 so that condition 19 becomes fully computationnaly 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 Δ can be seen as a composition of three adjacent intervals: Δ_1,Δ_2 and Δ_3. Interval Δ_2 spans over the time from the first to the last hyperperiod point in Δ, whereas Δ_1 and Δ_3 cover the pieces of execution occurring respectively before and after Δ_2. Figure 6 illustrates the above described Δ decomposition.

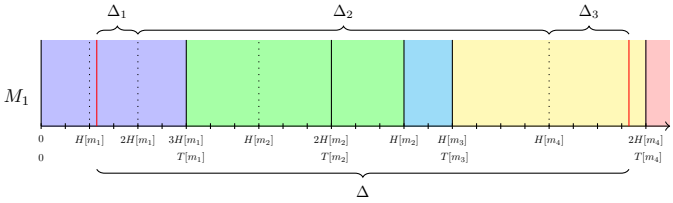


Fig. 6: Decomposition of Δ into three intervals.

The maximum value of demand function in any interval Δ_1 for a module M_j, denoted by $df_{M_j}(\Delta_1)$, is given by:

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

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 Δ_3, denoted by $df_{M_j}(\Delta_3)$, can be found:

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

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 for any interval Δ_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 Δ, is denoted by $df_{M_j}(\Delta_2)$ such that:

$$\max df_{M_j}(\Delta_2) \leq \Delta \max_{m_k} U(m_k) \tag{20}$$

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 Δ and any valid mode time δ_k in mode m_k from module M_j the following relation is derived:

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

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 Δ for which the second condition of Theorem 2 is violated:

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

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

$$\Delta < \frac{2 \times \sum_{M_j \in Modules} \max_{m_k}(U(m_k)H[m_k])}{1 - \sum_{M_j \in Modules} \max_{m_k} U(m_k)} \tag{21}$$

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.

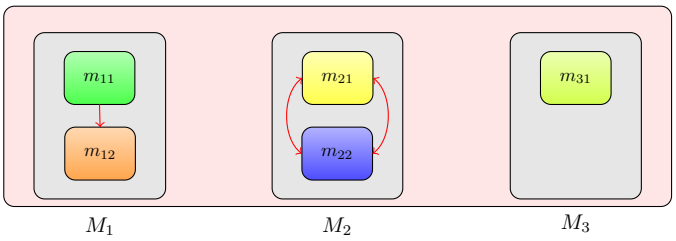


Fig. 7: A 3-modules E-TDL system and its mode switches

Module	mode	T	T_{sw}	tasks
M_1	m_{11}	10	10	$\tau_{111}(0,2,6,10), \tau_{112}(0,1,5,5)$
	m_{12}	8	-	$\tau_{121}(2,1,2,8)$
M_2	m_{21}	4	4	$\tau_{211}(0,1,4,4)$
	m_{22}	8	8	$\tau_{221}(1,1,1,8)$
M_3	m_{31}	8	-	$\tau_{311}(2,1,1,8)$

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 $mdbf_{M_1}$,

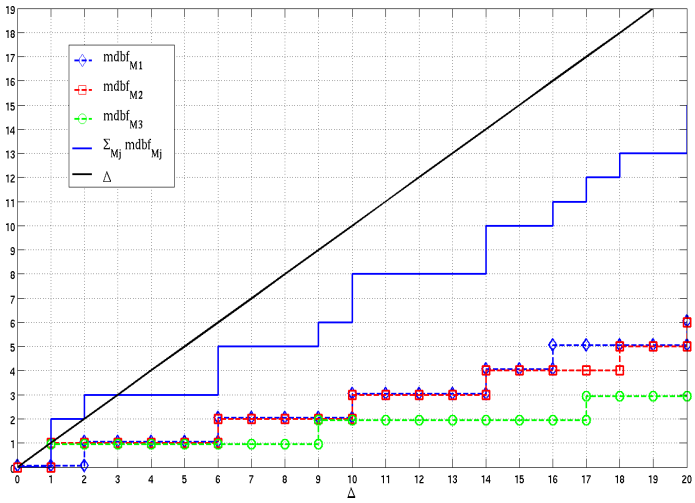


Fig. 8: Maximal demand functions

$mdbf_{M_2}$, $mdbf_{M_3}$, and their sum which is compared to the length of the interval Δ. 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 $\sigma_{M_2} = (m_{22}, 1, 2, \emptyset, \emptyset, 0)$ and $\sigma_{M_3} = (m_{31}, 2, 3, \emptyset, \emptyset, 0)$ 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 $\sigma_{M_1} = (m_{12}, 2, 4, \emptyset, \emptyset, 0)$, $\sigma_{M_2} = (m_{22}, 1, 3, \emptyset, \emptyset, 0)$, $\sigma'_{M_2} = (m_{22}, 0, 2, \emptyset, \emptyset, 0)$, $\sigma_{M_3} = (m_{31}, 2, 4, \emptyset, \emptyset, 0)$, and $\sigma'_{M_3} = (m_{31}, 1, 3, \emptyset, \emptyset, 0)$ 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 $((m_1, \delta_1), (m_{22}, 1), (m_{31}, 2))$ where (m_1, δ_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 $\Delta = 1$).

	path	gcd
p_{12}	$m_{11} \rightarrow m_{12}$	2
p_{22}	$m_{21} \rightarrow m_{22}$	4
p_{31}	m_{31}	8

Tab. 2: Greatest common divisor of paths

The greatest common divisors $\gcd(p_a \cup p_b)$ 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 $\gcd(p_{22} \cup p_{31}) = 4$. From Lemma 2 we obtain

	p_{12}	p_{22}	p_{31}
p_{12}	-	2	2
p_{22}	2	-	4
p_{31}	2	4	-

Tab. 3: Greatest common divisors for pairs of paths leading to the modes that can execute concurrently

$\alpha(p_{22}, p_{31}) = \{0, 1\}$. Mode m_{31} can start either at the same time as m_{22} ($0 \times \gcd(p_{22} \cup p_{31})$) either 4 time units before ($1 \times \gcd(p_{22} \cup p_{31})$). Equation (18) is not satisfied by any coefficient $\alpha(p_{22}, p_{31})$ for module M_2 state $(m_{22}, 1)$ and module M_3 state $(m_{31}, 2)$, since $(2-1+8) \bmod 8 \notin \{0, 4\}$. A parallel configuration where these two module states coincide is not observable and therefore the execution traces σ_{M_2} and σ_{M_3} 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:

$$\{((m_{12}, 2), (m_{22}, \delta_{22}), (m_{31}, \delta_{31})) \mid \delta_{22} \in \{0, 1\}, \delta_{31} \in \{1, 2\}\}$$

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 $(\alpha_2 \times \gcd(p_{12} \cup p_{22}))$ 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 (α_2, α_3). Each tuple (α_2, α_3) describes a mode activation pattern in which m_{22} and m_{31} started respectively $\alpha_2 \times \gcd(p_{12} \cup p_{22})$ and $\alpha_3 \times \gcd(p_{12} \cup p_{31})$ instants before m_1. Table 4 shows all valid tuples (α_2, α_3) together with possible walks starting at the initialization of every module and resulting in the activation pattern reflected by this tuple.

(α_2, α_3)	example walk		
	w_{12}	w_{22}	w_{31}
$(0, 0)$	$(m_{11}, 4)$	$(m_{21}, 10)$	$(m_{31}, 5)$
$(0, 2)$	$(m_{11}, 2)$	$(m_{21}, 1), (m_{22}, 2)$	$(m_{31}, 2)$
$(1, 1)$	$(m_{11}, 1)$	$(m_{21}, 2)$	$(m_{31}, 1)$
$(1, 3)$	$(m_{11}, 3)$	$(m_{21}, 5), (m_{22}, 1)$	$(m_{31}, 3)$
$(2, 0)$	$(m_{11}, 4)$	$(m_{21}, 5), (m_{22}, 2)$	$(m_{31}, 5)$
$(2, 2)$	$(m_{11}, 2)$	$(m_{21}, 2), (m_{22}, 1)$	$(m_{31}, 2)$
$(3, 1)$	$(m_{11}, 1)$	$(m_{21}, 1)$	$(m_{31}, 1)$
$(3, 3)$	$(m_{11}, 3)$	$(m_{21}, 4), (m_{22}, 1)$	$(m_{31}, 3)$

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

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).

	(0,0)	(0,2)	(1,1)	(1,3)	(2,0)	(2,2)	(3,1)	(3,3)
δ_{22}	2	2	4	4	6	6	0	0
δ_{31}	2	6	4	0	2	6	4	0

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

[1] B. Andersson. Uniprocessor EDF Scheduling with Mode Change. In *Proceedings of the 12th International Conference on Principles of Distributed Systems*, OPODIS '08, pages 572–577, Berlin, Heidelberg, 2008. Springer-Verlag

[2] C. M. Bailey. *Hard real-time operating system kernel. Investigation of mode change. Task 14 Deliverable on ESTSEC Contract 9198/90/NL/SF* British Aerospace Systems Ltd., 1993

[3] O. Bordellès and V. Bordellès. *Arithmetic Tales*. Universitext Series. Springer, 2012.

[4] S. K. Baruah, L. E. Rosier, and R. R. Howell. Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor. *Real-Time Syst.*, 2(4):301–324, Oct. 1990.

[5] S. K. Baruah. Feasibility Analysis of Recurring Branching Tasks. In *ECRTS*, pages 138–145, 1998.

[6] G. C. Buttazzo. *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series)*. Springer-Verlag TELOS, Santa Clara, CA, USA, 2004.

[7] G. C. Buttazzo, L. Abeni, and S. S. S. Anna. Elastic Task Model For Adaptive Rate Control. In *IEEE Real-Time Systems Symposium*, pages 286–295, 1998

[8] A. Easwaran, M. Anand, and I. Lee. Compositional Analysis Framework Using EDP Resource Models. In *RTSS*, pages 129–138, 2007.

[9] E. Farcas. *Scheduling Multi-Mode Real-Time Distributed Components*. PhD Thesis, Department of Computer Science, University of Salzburg, July 2006.

[10] E. Farcas, C. Farcas, W. Pree, and J. Templ. Transparent Distribution of Real-Time Components Based on Logical Execution Time. In *LCTES*, pages 31–39, 2005.

[11] N. Fisher and M. Ahmed. Tractable Real-Time Schedulability Analysis for Mode Changes under Temporal Isolation. In *ESTImedia*, pages 130–139, 2011

[12] J. V. Z. Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[13] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: A Time-Triggered Language For Embedded Programming. In *EMSOFT*, pages 166–184, 2001.

[14] C. M. Kirsch and A. Sokolova. The Logical Execution Time Paradigm. In *Advances in Real-Time Systems*, pages 103–120, 2012.

[15] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM*, 20(1):46–61, 1973.

[16] N. F. Martinek and W. Pohlmann. Mode Switching in GIA – An ADA Based Real-Time Framework. Department of Scientific Computing, University of Salzburg.

[17] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. PhD Thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, USA, 1983.

[18] P. Pedro. *Schedulability of Mode Changes in Flexible Real-Time Distributed Systems*. PhD Thesis, Department of Computer Science, University of York, September 1999.

[19] P. Pedro and A. Burns. Schedulability Analysis for Mode Changes in Flexible Real-Time Systems. In *ECRTS*, pages 172–179, 1998.

[20] R. Pellizzoni and G. Lipari. Feasibility Analysis of Real-Time Periodic Tasks with Offsets. *Real-Time Systems*, 30(1-2):105–128, 2005.

[21] J. Ramírez-Alfonsín. Complexity of the Frobenius problem. *Combinatorica*, 16:143–147, 1996.

[22] J. Real and A. Crespo. *Mode Change Protocols for Real-Time Systems: A Survey and a New Proposal*. Real-Time Syst., 26(2) :161–197, March 2004

[23] L. Santinelli, G. C. Buttazzo, and E. Bini. Multi-Moded Resource Reservations. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 37–46, 2011.

[24] J. Shallit. The Frobenius Problem and Its Generalizations. In M. Ito and M. Toyama, *Developments in Language Theory*, volume 5257 of *Lecture Notes in Computer Science*, pages 72–83. Springer Berlin Heidelberg, 2008.

[25] N. Stoimenov, S. Perathoner, and L. Thiele. Reliable Mode Changes in Real-Time Systems with Fixed Priority or EDF Scheduling. In *Proceedings of Design, Automation and Test in Europe, 2009* , pages 99–104, Apr 2009. IEEE.

[26] J. Templ. Timing Definition Language (TDL) Specification 1.5 Technical Report T024, Department of Computer Science, University of Salzburg, Austria, October 2008.

[27] L. Thiele, S. Chakraborty, and M. Naedele. *Real-Time Calculus for Scheduling Hard Real-Time Systems*. In *The 27th Annual International Symposium on Computer Architecture(ISCA)*, volume 4, pages 101 –104 vol.4, 2000

[28] K. Tindell and A. Alonso. *A very simple protocol for mode changes in priority preemptive systems*. Technical report, Universidad Politécnica de Madrid, 1996

[29] K. Tindell, A. Burns, and A. Wellings. *Mode changes in priority preemptively scheduled systems*. In Proceedings of the Real Time Systems Symposium, pages 100–109, 1992