Reliable Mode Changes in Real-Time Systems with Fixed Priority or EDF Scheduling

Nikolay Stoimenov Simon Perathoner Lothar Thiele
Computer Engineering and Networks Laboratory
ETH Zurich, 8092 Zurich, Switzerland
Email: {nikolays, perathoner, thiele}@tik.ee.ethz.ch

***Abstract*—Many application domains require adaptive real-time embedded systems that can change their functionality over time. In such systems it is not only necessary to guarantee timing constraints in every operating mode, but also during the transition between different modes. Known approaches that address the problem of timing analysis over mode changes are restricted to fixed priority scheduling policies. In addition, most of them are also limited to simple periodic event stream models and therefore, they can not faithfully abstract the bursty timing behavior which can be observed in embedded systems. In this paper, we propose a new method for the design and analysis of adaptive multi-mode systems that supports any event stream model and can handle earliest deadline first (EDF) as well as fixed priority (FP) scheduling of tasks. We embed the analysis method into a well-established modular performance analysis framework based on Real-Time Calculus and prove its applicability by analyzing a case study.**

I. Introduction

Several application domains ask for real-time embedded systems that can adapt their behavior at run-time by changing their operating mode. Examples of multi-mode systems are adaptive control systems in the automotive domain or new mobile phone applications such as software defined radio receivers. A mode change in an embedded application can be requested for several reasons. For example the system might need to switch to an emergency state, to adapt its behavior to changed conditions in the environment or to change its resource usage.

Another example is the use of scheduling servers within an operating system. They assign a periodic computing service to each application and therefore, lead to a largely reduced timing interference between applications. But the computing bandwidth available to a specific application needs to be adapted to its needs and new servers appear with new applications. Each of these adaptations leads to a mode change.

We assume that a mode change can involve changes in the set of executed tasks, changes in the parameters of tasks (e.g. execution time, deadline) or changes in the activation pattern of tasks. In such adaptive real-time systems, all deadlines must be provably met not only in the individual operating modes, but also during the transitions between modes. It is therefore essential to provide designers of multi-mode real-time systems with appropriate instruments for the verification of timing constraints across mode changes.

In general, a sudden mode change at run-time can have severe and unexpected impacts on the timing behavior of a system. For instance, if the execution of a task is triggered by an event stream, replacing the event stream instantaneously with a less demanding one (e.g. one with a larger period) can nevertheless harm the system, because bursts of events appearing at the switching time can lead to a transient overload of the system and missed deadlines.

The problem of timing analysis across mode changes has been addressed previously, see [1], [2] and the references therein. An analysis approach for mode changes on single processor systems with rate-monotonic scheduling is introduced in [3]. The analysis approach is improved and extended to deadline-monotonic scheduling in [4]. The model is augmented with transition offsets in [5], which permits to avoid overload situations. However, a way to calculate such offsets is not provided. A slightly different mode change protocol is introduced in [1], together with an algorithm for offset calculation. All these analysis methods are limited to strictly periodic task activation. This restriction was recently overcome in [2], where the authors consider mode changes under a more general task activation scheme. In their model each event stream is described by three parameters: period, jitter and minimum distance between events. Although this model captures considerably more complex activation patterns, it still describes only a limited set of event streams. Mode changes for distributed systems are also considered in [2]. However, the authors limit their discussion to identifying recurring effects of a mode change as the main problem for the analysis of an example system, without devising a detailed analysis procedure for the general distributed case. The main limitation of all mentioned analysis approaches is the restriction to scheduling policies with fixed task priorities (FP). To the best of our knowledge, the analysis of mode changes under scheduling policies with dynamic priorities like earliest deadline first (EDF) have not been considered so far.

Contributions of this work: We present a method for timing analysis of single-processor multi-mode systems with earliest deadline first (EDF) or fixed priority (FP) scheduling of tasks that supports any task activation pattern. We show how the method can be applied to transform a non-schedulable mode change into a schedulable one using an offset. We prove the applicability of the presented method by analyzing a case study.

II. Mode Change – An Example

For illustrative purposes, we present a multimedia system that undergoes a mode change. It will serve as a case study throughout the paper, and will help us to visualize the presented theoretical results. Fig. 2 shows the architecture of a digital set-top box implementing a picture-in-picture (PiP) application where two concurrent MPEG-2 video streams are being decoded on a single CPU and displayed on the same

Work supported in part by the Swiss National Science Foundation (grant 200020-116594), by the National Center of Competence in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation (grant 5005-67322), and by the European Community's Seventh Framework Programme FP7/2007-2013 PREDATOR (grant 216008).

978-3-9810801-5-5/DATE09 © 2009 EDAA

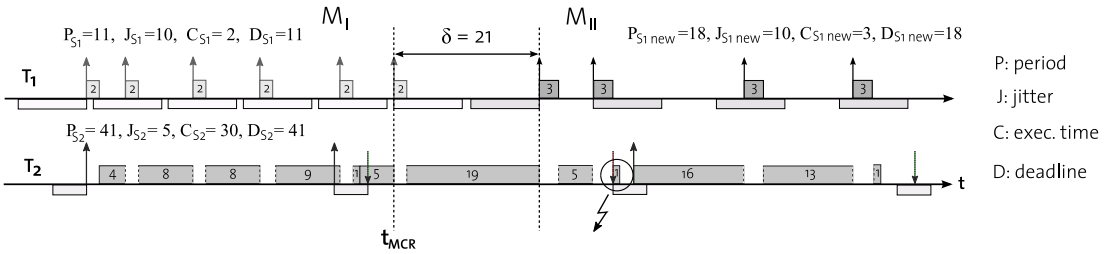


Fig. 1. Missed deadline due to transient overload during mode change

output device. The streams corresponding to the main window W_1 and small window W_2 in the output device are denoted as S_1 and S_2, respectively. The CPU executes two tasks T_1 and T_2 which perform the MPEG-2 decoding for S_1 and S_2, respectively. The inputs to the tasks are compressed bitstreams and their outputs are decoded macroblocks, which are written into playout buffers.

Consider now that the video displayed on W_1 changes from video stream S_1 (mode I) to a video stream $S_{1\text{ new}}$ with a lower workload (mode II). Such an instantaneous mode change is likely to result in a transient overload of the system and missed deadlines, which in this scenario translates to distorted playback. A common way to reduce the workload during mode changes is to delay the start of mode II by an offset [5]. In this case workload from mode I is accepted up to the time of the mode change request t_{MCR}, while workload from mode II is not accepted before $t_{\text{MCR}} + \delta$, where δ is the length of the offset. The designer of the set-top box needs to find an offset of sufficient length in order to avoid distorted images during stream changes. At the same time, the offset should be kept as short as possible. Adding an offset delays the switch to the new video stream however, the disruption is completely predictable at design time.

Choosing the offset for a mode change is not trivial at all. To illustrate this, consider that the two tasks T_1 and T_2 are scheduled according to a preemptive FP scheduling policy, with T_1 having higher priority than T_2. For the sake of simplicity, assume that T_1 and T_2 are triggered by periodic event streams with jitter and have constant execution times. In Section VII we will analyze the set-top box mode change scenario for realistic MPEG-2 video streams. For now, consider the stream parameters reported in Fig. 1. A reasonable guess for the length of the offset at the mode change is $\delta = 21$, which is the maximum distance between the arrival of two events in stream S_1. Since the workload for $S_{1\text{ new}}$ is slightly smaller ($3/18 < 2/11$), one might assume that this offset is sufficient for meeting all deadlines. However, the trace depicted in Fig. 1 shows that with $\delta = 21$ an activation of task T_2 can miss its deadline due to the video change in the higher priority stream. In other words, switching between two video streams in W_1 can cause an unpredictable disruption in the video playback of W_2. The example shows that formal methods are desirable for the design and analysis of multi-mode real-time systems.

III. Real-Time Calculus

In this section we describe the framework of Modular Performance Analysis with Real-Time Calculus (in short called MPA-RTC), on top of which we will develop the analysis of mode changes. MPA-RTC is a compositional framework for system level performance analysis of distributed real-time systems [6]. It has its roots in Network Calculus [7], [8].

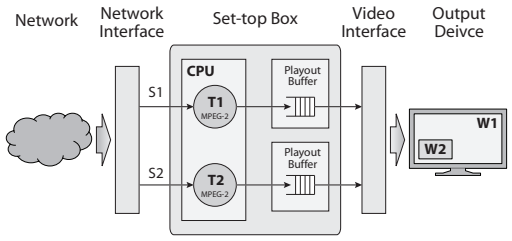


Fig. 2. System architecture

It analyzes the flow of event streams through a network of processing and communication resources in order to compute worst-case backlogs, end-to-end delays and throughput.

1) A General Event Stream Model: Event streams are abstracted by a tuple $\alpha(\Delta) = [\alpha^u(\Delta), \alpha^l(\Delta)]$ of upper and lower arrival curves which provide an upper and a lower bound on the number of events in *any* time interval of length Δ. If $R[s,t)$ denotes the number of events that arrive in the time interval $[s,t)$, then the following inequality holds

$$\alpha^l(t-s) \leq R[s,t) \leq \alpha^u(t-s) \quad \forall s < t,$$

where $\alpha^u(\Delta) = \alpha^l(\Delta) = 0$ for $\Delta \leq 0$. Arrival curves substantially generalize conventional event stream models such as sporadic, periodic or periodic with jitter. Note that often the domain of arrival curves are workload units. Event-based arrival curves can be converted to workload-based arrival curves by scaling with the best-case/worst-case execution demand of events. We will use the workload-based interpretation in the rest of the paper.

2) A General Resource Model: The availability of processing or communication resources is described by a tuple $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ of upper and lower service curves which provide an upper and a lower bound on the available service in *any* time interval of length Δ. The service is expressed in an appropriate workload unit compatible to that of the arrival curve, like number of cycles for computing resources or bytes for communication resources. If $C[s,t)$ denotes the amount of workload units available from a resource in the time interval $[s,t)$, then the following inequality holds

$$\beta^l(t-s) \leq C[s,t) \leq \beta^u(t-s) \quad \forall s < t.$$

3) Processing Model and Analysis: In real-time systems, event streams are typically processed by a sequence of HW/SW components. In the framework of MPA-RTC such processing or communication components are modeled by abstract performance components that act as curve transformers in the domain of arrival and service curves, where the transfer function depends on the modeled processing semantics. A typical example for an abstract performance component in the context of MPA-RTC is a *Greedy Processing Component* (GPC), shown in Fig. 3(a). It models a task that is triggered

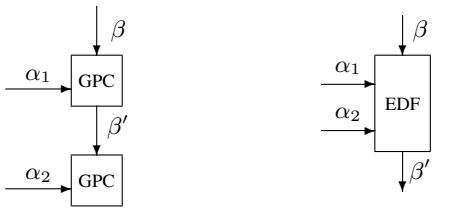


Fig. 3. Two abstract components in MPA-RTC

by the events of the incoming event stream which queue up in a FIFO buffer. The task processes the events in a greedy fashion, while being restricted by the availability of processing resources. The worst-case response time experienced by an event at a GPC and the worst-case backlog of a GPC satisfy the following upper bounds

$$\sup_{\lambda \geq 0} \left\{ \inf\{\tau \geq 0 : \alpha^u(\lambda) \leq \beta^l(\lambda + \tau)\} \right\} \stackrel{def}{=} \text{Del}(\alpha^u, \beta^l), \quad (1)$$

$$\sup_{\lambda \geq 0} \{\alpha^u(\lambda) - \beta^l(\lambda)\} \stackrel{def}{=} \text{Buf}(\alpha^u, \beta^l). \quad (2)$$

Having a delay requirement d for a GPC component means that we require $\text{Del}(\alpha^u, \beta^l) \leq d$ for $\forall \Delta \in \mathbb{R}^{\geq 0}$ which can be expressed as the inequality

$$\alpha^u(\Delta - d) \leq \beta^l(\Delta) \quad \forall \Delta \in \mathbb{R}^{\geq 0}. \quad (3)$$

A GPC is said to be schedulable, if for all events of the triggering event stream the above inequality is satisfied.

Performance components are connected to a network which reflects the data flow as well as the hardware architecture of the modeled system. Some scheduling policies for shared resources are expressed by the way abstract resource streams β are distributed among the different abstract components. For instance, in a preemptive FP scheduling scheme with two tasks the lower priority task only gets the resources left over by the higher priority task. In the MPA-RTC framework this is modeled by connecting the service stream output β' of the higher priority GPC with the service stream input of the lower priority GPC as shown in Fig. 3(a). We find a lower bound for β' with

$$\beta'^l(\Delta) = \sup_{0 \leq \lambda \leq \Delta} \{\beta^l(\lambda) - \alpha^u(\lambda)\}. \quad (4)$$

Other scheduling policies are modeled by means of abstract performance components with multiple event stream inputs and tailored internal relations. An example is the abstract performance component for EDF scheduling, depicted in Fig. 3(b) for the case of two tasks. For a general EDF component with n tasks, the deadlines of all triggering event streams are guaranteed to be met if the total maximum processing demand Ω of all the streams is smaller than the minimum service β^l available to the component. This is the case if the following inequality holds

$$\Omega(\Delta) = \sum_i^n \alpha_i^u(\Delta - d_i) \leq \beta^l(\Delta) \quad \forall \Delta \in \mathbb{R}^{\geq 0}, \quad (5)$$

where d_i is the deadline associated to event stream i and the function $\alpha_i^u(\Delta - d_i)$ is the maximum processing demand of stream i which results from events that arrive and have their deadlines within a time interval of size Δ. This function is often referred to as *demand bound function* [9]. We can also compute the remaining service from an EDF component

$$\beta'^l(\Delta) = \sup_{0 \leq \lambda \leq \Delta} \{\beta^l(\lambda) - \sum_i^n \alpha_i^u(\lambda)\}. \quad (6)$$

This allows us to connect GPC and EDF components using their service inputs and outputs, and model arbitrarily complex hierarchically scheduled systems where FP and EDF scheduling policies are mixed.

IV. Mode Change Model

In this section we define a model for mode changes which provides the basis for the timing analysis described in the following sections. A *mode change request* (MCR) can be initiated by the environment or by the system. An MCR is asynchronous and can happen at any time of a mode execution denoted by t_{MCR}. In order to exclude interference of multiple mode changes, we assume that a new MCR cannot occur during a transition between modes. We will refer to the mode in which a change is initiated as mode I. The target mode of a change will be called mode II. Each mode comprises a set of tasks to execute. The task set or the parameters of single tasks can change only at mode switches. Each task is associated with an activation stream expressed as an arrival curve α, a best-case execution demand b, a worst-case execution demand w, and a deadline d. These parameters are defined for all modes however, they may be different for the different modes. During a mode change, we differentiate between several types of tasks. *Added tasks* are active in mode II, and inactive in mode I. Therefore, we have $\alpha_I(\Delta) = 0 \ \forall \Delta$. Activations for these tasks will be accepted only at time $t \geq t_{\text{MCR}}$. *Completed tasks* are active in mode I, and inactive in mode II, thus $\alpha_{II}(\Delta) = 0 \ \forall \Delta$. Activations for these tasks will be accepted only at time $t < t_{\text{MCR}}$. For all activations the task execution needs to be completed, even if this happens after t_{MCR}. *Unchanged tasks* are active in mode I and mode II with the same parameters. An MCR does not affect them. *Changed tasks* are active in mode I and mode II with different parameters $\alpha_I \neq \alpha_{II} \vee w_I \neq w_{II} \vee b_I \neq b_{II} \vee d_I \neq d_{II}$. They are activated with the first set of parameters for $t < t_{\text{MCR}}$ and with the second set of parameters for $t \geq t_{\text{MCR}}$. As for the completed tasks, changed tasks need to complete all executions started before the MCR, even if this happens after t_{MCR}.

The potential transient overlap in the workload deriving from tasks of modes I and II after the MCR can lead to an overload situation for the system. As indicated in Section II, the overload can be avoided by delaying the start of tasks in mode II by an offset of length δ. In this case, activations for added tasks will be accepted only at time $t \geq t_{\text{MCR}} + \delta$. For changed tasks, no activations are accepted in the interval $[t_{\text{MCR}}, t_{\text{MCR}} + \delta)$. They are activated with the parameters of mode I for $t < t_{\text{MCR}}$ and with the parameters of mode II for $t \geq t_{\text{MCR}} + \delta$.

Schedulability for a system undergoing a mode change is defined as all tasks in the system *always* meeting their deadlines (in mode I, mode II and during the transition). In the next two sections we present methods for schedulability analysis across mode changes under FP and EDF scheduling, respectively. In both cases we assume schedulability for both modes in mutual exclusion.

V. Mode Change for Fixed Priority Scheduling

Consider a single processor system which executes multiple tasks according to a preemptive FP scheduling policy. Consider two modes I and II in which the system is schedulable, and a change from mode I to mode II for which schedulability needs to be proven. Assume that for a generic task τ a service curve $\tilde{\beta}^l$ is given as input that lower bounds the service which is available to τ for *all* time intervals, i.e. it is valid in mode I, mode II, and during the transition. If τ is an unchanged task with activation pattern α and deadline d, using (3) we can check for schedulability, and with (4) we can directly compute bounds on the remaining service $\tilde{\beta}'$.

If τ is a changed task, we have to take into account that its workload changes from α_I to α_{II} and its deadline from d_I to d_{II}. We want to find an arrival curve $\tilde{\alpha}^u$ which upper bounds the workload of τ for *all* time intervals. Given $\tilde{\alpha}^u$ we will compute the service remaining for lower priority tasks by means of (4). Here we consider two different cases. In the first case we assume that the switch from mode I to mode II is immediate. In the second case we consider an offset of length δ between the two modes. Note that added and completed tasks can be treated as changed tasks.

Case 1: Immediate Start of Mode II Task: In this case, mode I activations of τ stop at t_{MCR} and mode II activations are accepted for $t \geq t_{\text{MCR}}$. In order to guarantee the schedulability of τ, we have to verify that the task always meets its deadlines. For events of mode I the deadline is guaranteed to be met if $\text{Del}(\alpha_I^u, \tilde{\beta}^l) \leq d_I$. For events of mode II, we have to consider not only the worst-case activation pattern of mode II given by α_{II}^u, but also the fact that all buffered events of mode I need to be processed before any event of mode II. Thus, all events of mode II are guaranteed to always meet their deadline if $\text{Del}(\alpha_{II}^u + \text{Buf}(\alpha_I^u, \beta_I^l), \tilde{\beta}^l) \leq d_{II}$.[1]

In order to safely bound the service remaining for lower priority tasks, we need to find an arrival curve $\tilde{\alpha}^u$ that upper bounds the workload of τ for *all* time intervals, i.e. which is valid in mode I, mode II, and also during the transition. Such an arrival curve can be computed with

$$\tilde{\alpha}^u(\Delta) = \sup_{0 \leq \lambda \leq \Delta} \{\alpha_I^u(\Delta - \lambda) + \alpha_{II}^u(\lambda)\}. \tag{7}$$

This result can be understood by looking at time intervals of length Δ that start before the mode change and end after it. In terms of the resulting arrival curve we need to consider the worst case with respect to *any* location of this time interval relative to the mode switch, i.e. starting $\Delta - \lambda$ time units before and ending λ time units after.[2]

Case 2: Delayed Start of Mode II Task: In this case, mode I activations of τ stop at t_{MCR}, while mode II activations are accepted only for $t \geq t_{\text{MCR}} + \delta$. In order to guarantee schedulability, again we have to make sure that deadlines are always met. For events of mode I we have to check that $\text{Del}(\alpha_I^u, \tilde{\beta}^l) \leq d_I$. For events of mode II, we have to consider not only the existence of events remaining from mode I, but also that some of them could be processed during the offset interval. Thus, all events of the mode II are guaranteed to meet their deadline if $\text{Del}(\alpha_{II}^u + \text{Buf}(\alpha_I^u, \beta_I^l) - \tilde{\beta}^l(\delta), \tilde{\beta}^l) \leq d_{II}$.

[1] The sum of an arrival curve α and a constant C is an arrival curve α_S defined as $\alpha_S(\Delta) = \alpha(\Delta) + C$ for $\Delta > 0$ and $\alpha_S(\Delta) = 0$ otherwise.

[2] For the detailed proofs of (7), (8), (9), (10) and their graphical representations, we refer the reader to [10].

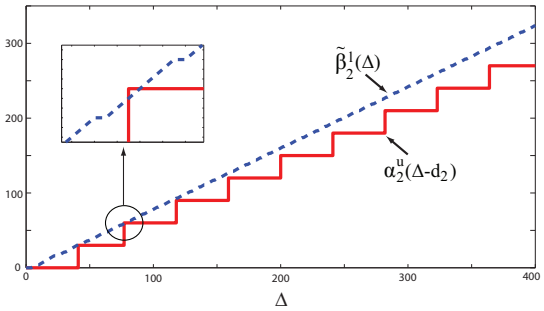


Fig. 4. Insufficient remaining service for T_2

The workload of τ is safely upper bounded by the following equation which is valid in mode I, mode II, and also during the transition

$$\tilde{\alpha}^u(\Delta) = \max\left\{\alpha_{II}^u(\Delta), \sup_{0 \leq \lambda \leq \Delta} \{\alpha_I^u(\Delta - \lambda) + \alpha_{II}^u(\lambda - \delta)\}\right\}. \tag{8}$$

In comparison to (7), we just shift the arrival curve associated to the second mode by the offset δ. In addition, we need to explicitly consider the arrival curve of mode II; in (7) it is implicitly taken into account.

Let us consider again the application scenario presented in Section II. We want to determine a length for the offset δ which is sufficient to guarantee all timing constraints for the video decoding. First we represent the arrival pattern of S_1, $S_{1\text{ new}}$ and S_2 with appropriate arrival curves $\alpha_{1,\text{I}}$, $\alpha_{1,\text{II}}$ and α_2, respectively. This is simple due to the periodic nature of the event streams. Then, we compute bounds for the workload of T_1 which are valid for all time intervals by means of (8). At this point we can safely lower bound the remaining service $\tilde{\beta}_2$ for T_2 with (4). In order to verify whether it is sufficient for the timely decoding of video stream S_2, we use (3) with $\tilde{\beta}_2^{\,l}$ and α_2^u. For the described mode change with an offset of 21 time units the inequality is not satisfied, as highlighted in the plot of Fig. 4. This confirms the deadline miss observed in Fig. 1.

VI. Mode Change for EDF Scheduling

Consider a single processor system with EDF scheduling of multiple tasks and two modes, I and II, in which the system is schedulable. Consider a change from mode I to mode II for which schedulability needs to be proven. All deadlines are guaranteed to be met if for all time intervals the total maximum processing demand is always smaller than the minimum service β^l available from the processor. In order to verify this condition, we need to upper bound the total processing demand $\tilde{\Omega}$ for *all* intervals, i.e. we need to compute a bound which is valid for mode I, mode II, and also during the transition.

Let Γ_U be the set of unchanged tasks and Γ_C the set of completed, added and changed tasks. For the sake of simplicity, we replace each changed task with an equivalent pair of a completed and an added task, such that Γ_C consists of completed and added tasks only. The maximum processing demand of the task set Γ_U will be denoted by Ω_U and can be computed with (5). Like for the FP case, for the tasks in Γ_C we distinguish two situations: an immediate switch between the modes, and a switch with an offset of size δ between the modes.

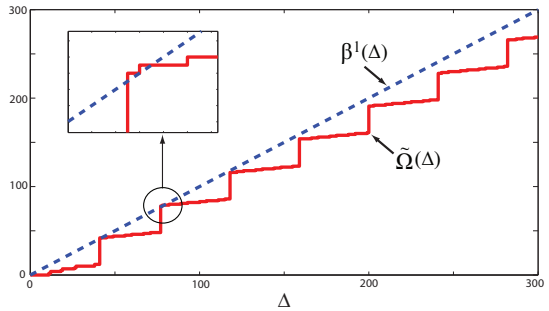


Fig. 5. Maximum processing demand exceeds minimum service

Case 1: Immediate Start of Mode II Tasks: Let Γ_{I} be the set of completed tasks and Γ_{II} the set of added tasks, with $\Gamma_C = \Gamma_{\mathrm{I}} \cup \Gamma_{\mathrm{II}}$. An upper bound for the maximum processing demand of Γ_C which is valid for all intervals can be computed with

$$\Omega_C = \sup_{0 \le \lambda \le \Delta} \Big\{ \sum_{\tau \in \Gamma_{\mathrm{I}}} \alpha^u_\tau(\Delta - \max\{d_\tau, \lambda\}) + \sum_{\tau \in \Gamma_{\mathrm{II}}} \alpha^u_\tau(\lambda - d_\tau) \Big\}. \tag{9}$$

This result can be explained by looking at time intervals of length Δ that include the mode switch. In contrast to (7), we need to consider the maximum combined *processing demand* of completed and added tasks for any position of the interval relative to the mode switch. In particular, we move the interval to the right by λ time units and take the worst case location. The *demand* is computed by shifting the arrival curves by the respective deadlines d_τ. For mode I, we have to consider that the processing demand of a completed task τ with deadline d_τ does not change as long as $\lambda \le d_\tau$. It only starts to reduce when $\lambda > d_\tau$ which leads to the shift of the respective arrival curve by $\max\{d_\tau, \lambda\}$.

Case 2: Delayed Start of Mode II Tasks: For a mode change with an offset of length $\delta > 0$ between the modes, the processing demand of Γ_C is upper bounded by

$$\Omega_C = \max \Bigg\{ \sum_{\tau \in \Gamma_{\mathrm{II}}} \alpha^u_\tau(\Delta - d_\tau), \sup_{0 \le \lambda \le \Delta} \Big\{ \sum_{\tau \in \Gamma_{\mathrm{I}}} \alpha^u_\tau(\Delta - \max\{d_\tau, \lambda\}) + \sum_{\tau \in \Gamma_{\mathrm{II}}} \alpha^u_\tau(\lambda - d_\tau - \delta) \Big\} \Bigg\}. \tag{10}$$

Similar to the relation between (7) and (8), we just shift the arrival curve of the second mode in (9) by an additional term δ which corresponds to the length of the offset. Again, we need to explicitly consider the arrival curve of mode II; in (9) it was implicitly taken into account.

For both cases all deadlines are guaranteed to be met despite the mode change if

$$\tilde{\Omega} = \Omega_U(\Delta) + \Omega_C(\Delta) \le \beta^l(\Delta) \quad \forall \Delta \in \mathbb{R}^{\ge 0}. \tag{11}$$

Let us come back to the application scenario of Section II. We want to determine whether the mode change is feasible without any offset for tasks T_1 and T_2 with EDF scheduling. We compute the maximum processing demand Ω_2 for T_2 with (5). Then, we use (9) to compute a safe upper bound for the processing demand Ω_1 of T_1 which is valid for all time intervals. At this point we can verify whether the total maximum processing demand $\tilde{\Omega}$ exceeds the minimum service β^l provided by the processor. Fig. 5 represents the two curves and shows that this is the case, i.e. a safe mode change cannot be guaranteed without an offset.

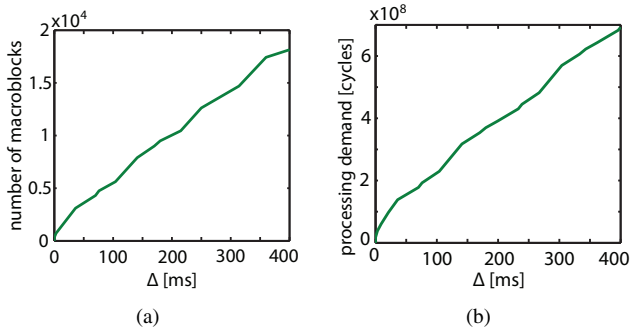
VII. Case Study

In this section we show how the proposed theory can be applied to the analysis of the system described in Section II with realistic MPEG-2 video streams. We consider that a mode change occurs in one of the video streams and we want to reduce its effects on the other video stream. The case study illustrates that the proposed methods can be applied to the mode change analysis of tasks scheduled with fixed or dynamic priorities and activated by streams with arbitrary complex arrival patterns. We show that a designer can find a sufficient offset such that a mode change in certain tasks does not affect the timing requirements of other tasks. In the case study, first we run a simulation with different sets of video streams and collect data for their arrival patterns and workload demands, then we compute workload arrival curves which we use in the mode change analysis.

Experimental setup: We simulate the system architecture shown in Fig. 2 with two different sets of video clips. The first set consists of regular clips with moderate to high motion content. It is selected to be representative for clips having high workload for the CPU. The second set consists of clips displaying still images. It is representative for low workload clips. All of the clips are MPEG-2 encoded and have the same frame resolution of 704 x 576 pixels. Each frame is made up of 1584 macroblocks. The down-scaling for the small PiP window is being done at the output device. The two video streams arrive at a constant bitrate of 8 Mbps and the two playout buffers are read at a constant rate of 25 frames/second. The CPU is set to run at a processor frequency of 3 GHz.

The two MPEG-2 decoding tasks have been mapped to the CPU. Both of them implement the variable length decoding (VLD), the inverse quantization (IQ), the inverse discrete cosine transform (IDCT), and the motion compensation (MC) tasks. A scheduler schedules the two tasks according to the selected QoS parameters for each stream. In this application the QoS has been reflected in the priorities of the tasks for the FP case or in the deadlines of the tasks for the EDF case. The simulation for the two sets of video clips have been performed using a SimpleScalar model [11] of the CPU (with the *sim-profile* configuration and the PISA instruction set). The video streams are modeled at the macroblock granularity. Each compressed macroblock in the input stream is made up of a variable number of bits. Therefore, at the macroblock granularity, a constant bit rate translates into a bursty arrival pattern. This can be observed in the upper arrival curve for the set of high workload video streams shown in Fig. 6(a) which is expressed in number of macroblocks arriving per time unit. Similarly, the number of processor cycles required to process a macroblock is also variable. During the simulation, workload data was gathered for the two tasks running the two different sets of streams. It represents the worst case resource demand in CPU cycles for processing one, two, and more consecutive macroblocks in a video stream. For the analysis, this data was combined with the arrival pattern of macroblocks in order to obtain workload based arrival curves. The corresponding curve for the high workload set is shown in Fig. 6(b).

Mode change analysis: The analysis starts from the fact that the system can meet all deadlines in all modes in mutual exclusion. Mode changes are first analyzed with offset $\delta = 0$.



(a) (b)

Fig. 6. Workload for high motion video in number of macroblocks (a) and processor cycles (b)

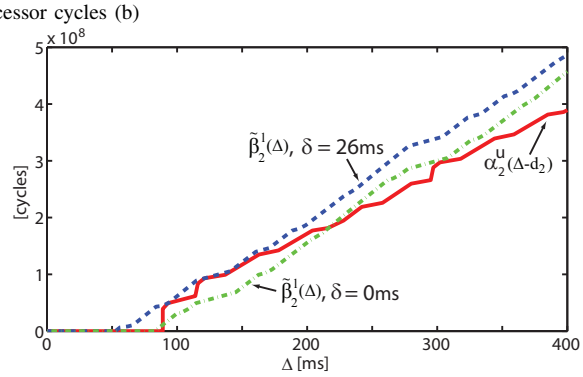


Fig. 7. Remaining service for stream S_2 with and without offset at mode change (FP scheduling)

If the system is found schedulable then the mode change can be performed safely without any offset. However, if the system is found unschedulable, the analysis is performed repetitively with different sizes of δ which are chosen by binary search. Analysis stops when the smallest δ is found that makes the system schedulable during the mode change.

Results: We use two different scenarios. In the first one the CPU scheduler implements a FP scheme, and in the second it implements EDF. Context switches are considered to take a negligible time. For both scenarios we perform a mode change where the video displayed on window W_1 changes from video stream S_1 (high motion content) to a video stream $S_{1\ new}$ with a lower workload (still image). We find an offset which is the smallest offset such that video stream S_2 meets its deadlines during the mode change, i.e. there is no disruption in the video displayed in window W_2.

Case A: Here we consider FP scheduling of the two MPEG-2 decoding tasks where stream $S_1/S_{1\ new}$ is processed with a higher priority than S_2. S_1 and $S_{1\ new}$ have both a maximum delay requirement of 14ms. In both modes, stream S_2 does not change. It has low workload and a delay requirement of 89ms. The analysis is first performed for an immediate change from S_1 to $S_{1\ new}$ without any offset. As can be seen in Fig. 7, when $\delta = 0$ the CPU service remaining for task T_2 which processes stream S_2 is not sufficient to meet the delay requirement. After repeating the analysis for various δ chosen by binary search, the smallest offset found is $\delta = 26$ms. The result is shown in Fig. 7.

Case B: This scenario is exactly the same as the previous one, except that the two tasks processing the two streams are scheduled by an EDF scheduler. Deadlines are chosen to be equal to the delay requirement for the previous case. Performing the same steps, we find a smallest offset $\delta = 12$ms such that T_2 always meets its deadline. Results are shown in

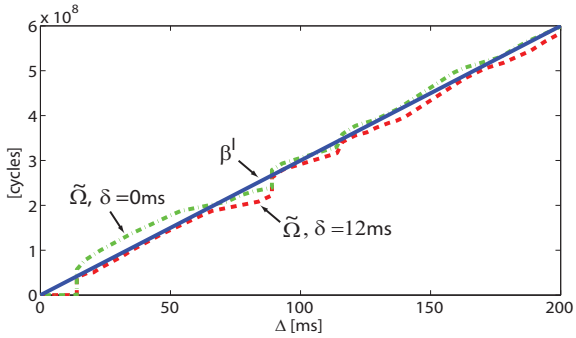


Fig. 8. Total maximum processing demand with and without offset at mode change (EDF scheduling)

Fig. 8.

Both scenarios show the counterintuitive result that performing a mode change can lead to timing violations during the transition period even when the system utilization is reduced in mode II.

VIII. Conclusion

In this paper we presented a new approach for the design and analysis of adaptive multi-mode embedded real-time systems. We introduced methods for scheduling analysis during mode transitions. They guarantee the timing behavior of multi-mode systems with FP or EDF scheduling of an arbitrary number of tasks. They can also be applied to any hierarchical combination of FP and EDF scheduling policies. The analysis is not bound to particular event stream models, but supports any arbitrary task activation pattern. We looked at immediate switches between modes, and showed that such changes often involve a transient overload of the system. Subsequently, we considered mode changes with an offset for the start of mode II tasks and we showed how the offset can be used to make an unschedulable transition schedulable. We proved the applicability of the presented methods through analysis of a case study.

References

[1] J. Real and A. Crespo, "Mode change protocols for real-time systems: A survey and a new proposal," *Real-Time Systems*, vol. 26, no. 2, pp. 161–197, 2004.

[2] R. Henia and R. Ernst, "Scenario aware analysis for complex event models and distributed systems," in *RTSS*, 2007, pp. 171–180.

[3] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham, "Mode change protocols for priority-driven preemptive scheduling," *Real-Time Systems*, vol. 1, no. 3, pp. 243–264, 1989.

[4] K. W. Tindell, A. Burns, and A. J. Wellings, "Mode changes in priority pre-emptively scheduled systems," in *RTSS*, 1992, pp. 100–109.

[5] P. Pedro and A. Burns, "Schedulability analysis for mode changes in flexible real-time systems," in *ECRTS*, 1998, pp. 172–179.

[6] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 4, pp. 101–104, 2000.

[7] R. Cruz, "A calculus for network delay, Parts 1 & 2," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 114–141, 1991.

[8] J. Y. Le Boudec and P. Thiran, *Network calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.

[9] S. Baruah, D. Chen, S. Gorinsky, and A. Mok, "Generalized multiframe tasks," *Real-Time Systems*, vol. 17, no. 1, pp. 5–22, 1999.

[10] S. Perathoner, N. Stoimenov, and L. Thiele, "Reliable mode changes in real-time systems with fixed priority or edf scheduling," Computer Engineering and Networks Laboratory, ETH Zurich, TIK Report 292, Sep. 2008.

[11] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," *IEEE Computer*, vol. 35, no. 2, pp. 59–67, 2002.