Dynamic budget management and budget reclamation for mixed-criticality systems

Gu, Xiaozhe; Easwaran, Arvind

2019

Gu, X., & Easwaran, A. (2019). Dynamic budget management and budget reclamation for mixed-criticality systems. *Real-Time Systems*, 55(3), 552-597.
doi:10.1007/s11241-019-09330-2

<https://hdl.handle.net/10356/143246>

<https://doi.org/10.1007/s11241-019-09330-2>

© 2019 Springer Science+Business Media. This is a post-peer-review, pre-copyedit version of an article published in *Real-Time Systems*. The final authenticated version is available online at: <http://dx.doi.org/10.1007/s11241-019-09330-2>

Downloaded on 26 Apr 2023 21:26:33 SGT

Dynamic Budget Management and Budget Reclamation for Mixed-Criticality Systems

Xiaozhe Gu · Arvind Easwaran

the date of receipt and acceptance should be inserted later

Abstract Many existing studies on mixed-criticality (MC) scheduling assume that low-criticality budgets for high-criticality applications are known apriori. These budgets are primarily used as guidance to determine when the scheduler should switch the system mode from low to high. Based on this key observation, in this paper we propose a dynamic MC scheduling model under which low-criticality budgets for individual high-criticality applications are determined at runtime based on a system-wide total low-criticality budget allocation for all the high-criticality applications combined. This total budget is used as guidance in our model to determine the need for a mode-switch. At run time, a job may terminate prior to its allocated low-criticality budget, and hence the remaining low-criticality budget is wasted. Therefore, in the paper, we also propose a budget reclamation scheme that can work along with the dynamic model to further prevent the occurrence of mode-switch.

Keywords Mixed-Criticality Scheduling · Dynamic Budget Allocation · Budget Reclamation

1 Introduction

An increasing trend in safety-critical real-time applications is that multiple functionalities with different levels of “criticality” (importance) are integrated together on a single computing platform [1]. To efficiently share the computing platform among those applications while ensuring isolation between different criticalities, Steve Vestal proposed the classic mixed-criticality (MC) task and scheduling model [2]. This task model is an extension of the standard sporadic real-time task system for two criticality levels. A task is defined as either a high-critical (HC) or

Xiaozhe Gu
Nanyang Technological University, SCSE, Singapore
E-mail: guxi0002@e.ntu.edu.sg

Arvind Easwaran
Nanyang Technological University, SCSE, Singapore
E-mail: arvinde@ntu.edu.sg

a low-critical (LC) task. A HC task τ_i has two execution time estimates C_i^L and C_i . While C_i is assumed to be greater than or equal to the worst case execution time (WCET) of the task, C_i^L is a lower estimate ($C_i^L \leq C_i$) that may not be sufficient for some jobs of the task. A LC task τ_i only has a single execution time estimate C_i that is assumed to be greater than or equal to its WCET.

Under the classic MC scheduling model, schedulability of HC tasks is assessed under the standard assumption that no task would execute beyond C_i . On the other hand, schedulability of LC tasks is only assessed under the assumption that each HC task τ_i would not execute beyond C_i^L . Consequently, a fundamental difference between this model and the standard non-MC scheduling model is that the scheduler can prioritize HC tasks over LC tasks when additional processing capacity ($> C_i^L$) is required for them. The system can then be seen as being in two different execution modes at runtime; LC mode as long as no job of any HC task τ_i executes beyond C_i^L , and HC mode thereafter during which no LC deadlines are required to be met.

From the above discussion we can observe that the execution estimate C_i^L is primarily used as a “budget” for the HC task τ_i in the LC mode. In fact, it helps the scheduler to determine whether the system should switch to the HC mode. Let B_i^L denote the maximum budget allocated to jobs of τ_i by an MC scheduler in the LC mode. Then, we can see that the classic MC scheduling model uses a static (fixed) budget allocation in the LC mode; $B_i^L = C_i^L$ for each HC task τ_i and $B_i^L = C_i$ for each LC tasks τ_i . Hence, we denote this model as the **static model**.

Motivation. In this paper we propose a more dynamic MC task and scheduling model (denoted in short as **dynamic model**) based on the following principle. *HC tasks can be allocated budgets in the LC mode dynamically at runtime depending on the overall processing requirements of the task system. As long as the allocated budgets continue to ensure schedulability requirements for all the tasks, it should be safe to do so.*

One important advantage of the dynamic model over the static one is that the application designers are not required to specify C_i^L any more, thus reducing their burden. But then a problem arises as to how a scheduler can safely determine mode-switch in the dynamic model so that schedulability of HC tasks is guaranteed at all times. To address this problem we propose **a technique that combines offline schedulability analysis with runtime budget allocation strategy** for HC tasks. Offline, we use schedulability analysis to determine a **total LC budget** allocation (single value) for all HC tasks combined. At runtime, we use a strategy to allocate budgets to individual jobs of HC tasks based on this total budget. Thus, the total LC budget computed offline is used as an indicator to determine mode-switch in the dynamic model, and hence the schedulability of HC tasks is ensured.

A system-wide budget for HC tasks in the LC mode, as opposed to budgets for individual tasks as in the static model, allows more flexibility at runtime to distribute this budget depending on execution requirements of individual jobs. It thus eliminates one of the pessimistic assumptions made in many existing MC studies, which is that when a single HC job has a high execution requirement (e.g., beyond C_i^L), all the other HC jobs in the system would also simultaneously have a high execution requirement. As a result, the mode-switch from LC to HC can also be potentially postponed, and more service is provided to LC tasks, because

many existing MC scheduling strategies either completely drop LC tasks or offer degraded service to them after the mode-switch.

To further delay the occurrence of mode-switch, we also propose a budget reclamation scheme which could work along with the dynamic budget allocation strategy, based on the observation that many jobs may complete prior to its allocated LC budget at runtime. With the budget reclamation scheme, jobs at runtime would first consume the unused LC budget from jobs with higher priority. As a result, HC jobs are less likely to consume their own LC budget, and the mode-switch will be less likely to be triggered.

Contributions. The contributions of this paper can be summarized as follows.

1. Dynamic Task and Scheduling model (Section 3.1): We propose a new MC task and scheduling model in which jobs of HC tasks are allocated budgets in the LC mode dynamically at runtime depending on their execution requirements. In this new model, it is also possible for LC tasks to receive guaranteed budgets after the system switches to HC mode.
2. EDF-UVD (Section 3.2): We propose a MC scheduling policy called Earliest Deadline First - Universal Virtual Deadlines (EDF-UVD), which is a variant of the well known EDF-VD [3] (Earliest Deadline First with Virtual Deadlines), except that in addition to HC tasks, even LC tasks have virtual deadlines under EDF-UVD.
3. Runtime Budget Allocation Strategy (Section 4): We propose a run time technique to distribute budgets to jobs of HC tasks in the LC mode, depending on the execution requirement of jobs as well as the total LC budget for HC tasks.
4. Budget Reclamation Scheme (Section 5): We propose a budget reclamation scheme that could work along with the proposed dynamic budget allocation strategy to further prevent the occurrence of mode-switch.
5. Schedulability Analysis (Section 6): We propose a schedulability analysis for our proposed dynamic model.

In Section 2, we introduce some related works in mixed-criticality scheduling theory that focus on supporting for the execution of low-criticality tasks. We propose some guidelines on how to determine the system-wide LC budget for all HC tasks combined in Section 7. In Section 8, we present the evaluation results about our proposed mechanism.

2 Related Work

The static model [2] is widely used in many previous studies for representing MC real-time workloads. This model requires multiple execution time estimates for different criticality levels in a system. Therefore a task may have up to five execution time estimates in a five-level MC system. As pointed out by Burns [4], it is an undue burden for the application designer to obtain such multiple execution time estimates. In [4], Burns also proposes a simplification to the static model, where each task only has two execution time estimates. By contrast, in this paper we propose using a single estimate from the application designer.

Many existing work on MC scheduling (e.g., [5–7, 3, 8, 9]) share the pessimistic strategy that all LC tasks will be immediately dropped once the system switches

to HC mode. Other studies have presented solutions to improve support for LC tasks [10–19]. These solutions can be broadly categorized into two classes.

1) The first category of studies [10, 11, 16, 12, 13, 19–23] support LC tasks by offering a degraded (and in some cases guaranteed) service to all of them when the system is in the HC mode. They do this either by reducing the dispatch frequency of jobs or by executing the LC tasks as background (low-priority) workload.

2) The second category of studies [15, 14, 18, 17] support LC tasks by offering a degraded service to only a subset of them, while keeping the service to others intact, depending on the specific HC tasks that demand additional execution. However, these studies do not provide any minimum guaranteed service to the LC tasks in the HC mode.

None of the above studies considered a dynamic budget allocation model for HC tasks such as the one proposed in this paper. Further, the LC service strategy we propose belongs to both the categories described above. It offers a minimum guaranteed service to all LC tasks in the HC mode (similar to some of the studies in the first category), and at the same time we are able to control the mode-switch depending on individual job execution requirements of HC tasks (similar to the second category). Thus, by combining both these approaches we are able to explore the trade-off between LC service guarantee on one hand and mode-switch on the other.

There are some other studies focusing on support for LC executions that are orthogonal to the problems addressed in this paper. Protocols [24, 25] are proposed to restore the system to LC mode after a mode-switch. Another study [26] uses the probability that C_i^L would be exceeded in the static model and derives corresponding schedulability analysis with permitted system failure probability.

3 Dynamic MC Task and Scheduling Model

3.1 Task model

In the dynamic MC task model, a LC task is defined as a tuple $(T_i, C_i, L_i = LC, \alpha_i)$ and a HC task is defined as a tuple $(T_i, C_i, L_i = HC)$, where T_i is the minimum separation time between successive job releases, L_i denotes the criticality level, and C_i upper bounds the worst case execution time of the task. Thus the application designer only needs to provide one execution time estimate C_i as in the standard non-MC task model.

For a LC task τ_i , its additional parameter $\alpha_i \in [0, 1]$ denotes proportion of utilization that is statically reserved for τ_i in HC mode. Thus, its HC budget is also fixed at $B_i^H = C_i \times \alpha_i$ ($\alpha_i \in [0, 1]$), while its LC budget is fixed at $B_i^L = C_i$. For a HC task τ_i , its HC budget is fixed at $B_i^H = C_i$ similar to the static model, but its LC budget is determined at runtime. It varies depending on the past execution demand of all HC jobs (details in Section 4).

We focus on implicit deadline task systems in this paper (i.e., relative deadline D_i is equal to T_i for each task), and consider the problem of scheduling n such tasks $\tau = \{\tau_1, \dots, \tau_n\}$ on a uniprocessor platform.

Similar to the static model, we can also define a schedulability criteria for algorithms that schedule task systems based on the dynamic model presented above. Since LC tasks also receive guaranteed budgets in HC mode, unlike the

static model, their deadlines cannot be ignored. Further, since individual HC tasks do not have a fixed budget in the LC mode, the scheduler only needs to ensure that each job of HC task τ_i receives as much execution as it needs up to C_i in both modes. If it is able to provide a budget of C_i for some job of τ_i while remaining in the LC mode, then it is free to do so. In fact, this flexibility in budget allocation is a key advantage of the dynamic model.

Definition 1 (MC-Schedulable) A task system τ in the dynamic model is defined to be **MC-Schedulable** by a scheduling algorithm if the following two conditions hold:

1. Jobs of LC task τ_i with deadline in the LC mode receive up to $B_i^L (= C_i)$ units of budget each, and jobs of LC task τ_i with deadline in the HC mode receive up to $B_i^H (= \alpha_i \times C_i)$ units of budget each.
2. Jobs of HC task τ_i receive as much budget as they need as long as it does not exceed C_i .

Finally, in the following table, we list some notations which will be used throughout the paper.

Table 1 Notations

τ^L	subset of LC tasks in τ
τ^H	subset of HC tasks in τ
$u_i = \frac{C_i}{T_i}$	the utilization of task τ_i
$U_L = \sum_{\tau_i \in \tau^L} u_i$	the utilization of task τ^L
$U_H = \sum_{\tau_i \in \tau^H} u_i$	the utilization of task τ^H
$J_{i,j}$	the j_{th} job of τ_i
$r_{i,j}$	release time of $J_{i,j}$
$d_{i,j}$	absolute deadline of $J_{i,j}$
$b_{i,j}$	the budget allocated to the j_{th} job of τ_i
$e_{i,j}$	the execution time of $J_{i,j}$, which is also budget consumed by $J_{i,j}$.
e_i^m	the maximum amount of time for which any job of τ_i has executed in the current busy interval

3.2 EDF-UVD Scheduling Strategy

We propose a new scheduling strategy for the dynamic model called EDF-UVD (Earliest Deadline First with Universal Virtual Deadlines). EDF-UVD is based on the well known algorithm EDF-VD [3] in which HC jobs are assigned virtual deadlines shorter than their original deadlines in the LC mode. These virtual deadlines ensure that when a mode-switch occurs, there is sufficient time for HC jobs to complete any additional execution before their original deadlines. The only difference between EDF-VD and EDF-UVD is that under EDF-UVD, even LC jobs would be assigned virtual deadlines shorter than their original deadlines in the LC mode. A job of LC task τ_i would be scheduled based on its virtual deadline as long as it has not executed for more than $\alpha_i \times C_i$ time units (i.e., execution proportional to its HC service level). Thereafter, it will be scheduled using its original deadline. Thus, EDF-UVD can be formally defined as follows.

1. Let $D_i(L) = x \times T_i$, where $x \in [0, 1]$, denote the virtual deadline for each task $\tau_i \in \tau$.
2. When the system is in the LC mode:
 - A job of HC task τ_i will be scheduled based on its virtual deadline $D_i(L)$.
 - A job of LC task τ_i that has executed for less than $\alpha_i \times C_i$ time units will be scheduled based on its virtual deadline $D_i(L)$.
 - A job of LC task τ_i that has executed for $\alpha_i \times C_i$ time units or more will be scheduled based on its original deadline D_i .
3. When the system is in the HC mode, both LC and HC jobs will be scheduled using their original deadlines.
4. In each mode, all jobs will be scheduled using the Earliest Deadline First policy.

EDF-UVD can reduce unnecessary budget allocations for LC tasks in comparison to EDF-VD. This happens when some job of a LC task τ_i executes beyond $\alpha_i \times C_i$ time units in the LC mode, and the system switches mode before the deadline of this job. In this case, any execution of the job beyond $\alpha_i \times C_i$ is unnecessary, because it is not required to be satisfied based on the definition of MC-schedulable. However, it is impossible to know prior to the mode-switch which execution is unnecessary. To address this issue, we use virtual deadlines for LC jobs in the LC mode. The main intuition behind this strategy is that the first $\alpha_i \times C_i$ execution units would be scheduled with a higher priority (based on virtual deadline), and the remaining execution units would be scheduled with a normal priority (based on original deadline). Thus, all jobs of LC tasks would prioritize the first $\alpha_i \times C_i$ execution units, thereby completing those execution units earlier than in the case of EDF-VD. Below we use a simple example to illustrate how EDF-UVD reduces unnecessary budget allocations.

Example 1 As shown in Figure 1(a), there are two LC jobs J_1^* and J_2^* that are released before t^* but have deadline after t^* , where t^* denotes the mode-switch instant. If J_1^* receives C_1 units of budget before t^* , then deadline miss would happen because J_2^* receives less than $\alpha_2 \times C_2$ units of budget before its deadline D_2 . However as shown in Figure 1(a), after J_1^* executes for $\alpha_1 \times C_1$ units of time, it is scheduled using its true deadline T_1 , and at that time J_2^* would have higher priority than J_1^* . As a result, the $(1 - \alpha_1) \times C_1$ units of budget previously consumed by J_1^* is now consumed by J_2^* , and the deadline miss at D_2 is avoided.

4 Maximum Execution-based Budget Allocation (MEBA)

In this section we present an efficient runtime strategy named MEBA (short for Maximum Execution-based Budget Allocation) for distributing the LC budget among jobs of HC tasks. Let β^* denote the service level of τ^H (subset of low-criticality tasks) in LC mode, which is equal to the proportion of U_H that is reserved as total budget for all HC tasks combined in the LC mode. While distributing the LC budget among jobs of HC tasks, we ensure that the total LC budget allocated to all HC tasks over any period of time is proportional to $\beta^* \times U_H^1$.

¹ In this and the following section (Section 5) we assume that the parameter β^* is known and present runtime budget allocation strategies for HC tasks. Later, in Section 6, we derive a schedulability condition for the dynamic MC task model, assuming it is scheduled under EDF-UVD and the runtime strategies presented in these sections. This condition can be used to determine a safe value for β^* that ensures the resulting system is MC-Schedulable.

Fig. 1 Example in which EDF-UV performs better than EDF-V

236 This strategy also determines when a mode-switch will occur in the dynamic
 237 model; it is precisely when the total demand of all HC tasks exceed $\beta^* \times U_H$. To
 238 be able to do this efficiently, we need to store some information about budgets
 239 consumed by HC jobs in the past. We first use a simple example to illustrate
 240 MEBA, and then present the general approach.

Fig. 2 Example illustrating the runtime budget allocation strategy

Example 2 As shown in Figure 2, a system $\tau = \{\tau_1, \tau_2, \tau_3\}$ is scheduled by EDF-UV and MEBA, where $\tau_1 = (10, 4, HC)$, $\tau_2 = (10, 4, HC)$, $\tau_3 = (10, 6, LC, \alpha_3 =$

0) and $\beta^* = 0.5$. Before $J_{1,1}$ begins execution at $r_{1,1} = 0$, we allocate a budget to it as follows.

$$\frac{b_{1,1}}{T_1} = \beta^* \times U_H \Rightarrow b_{1,1} = T_1 \times \beta^* \times U_H = 10 * 0.5 * 0.8 = 4.$$

Thus $J_{1,1}$ is allocated budget proportional to $\beta^* \times U_H$. When $J_{1,1}$ completes after executing for $e_{1,1} = 2$ time units, $e_{1,1}$ will be used to determine $b_{2,1}$, the budget for $J_{2,1}$. Thus, before $J_{2,1}$ begins execution at $t = 2$, we allocate a budget to it as follows.

$$\frac{e_{1,1}}{T_1} + \frac{b_{2,1}}{T_2} = \beta^* \times U_H \Rightarrow b_{2,1} = T_2 \times \left(\beta^* \times U_H - \frac{e_{1,1}}{T_1} \right) = 2$$

241 A fraction of budget equal to $e_{1,1}/T_1 = 0.2$ is reserved for future jobs of task τ_1 .
 242 The remaining budget of $(\beta^* \times U_H - e_{1,1}/T_1)$ is allocated to $J_{2,1}$.

When $J_{2,1}$ completes after executing for $e_{2,1} = 1$ time unit, $e_{2,1}$ will be used to determine $b_{1,2}$, the budget for $J_{1,2}$. As a result, at $r_{1,2} = 10$, a new value for $b_{1,2}$, which is now the budget for job $J_{1,2}$, will be computed as follows.

$$\frac{b_{1,2}}{T_1} + \frac{e_{2,1}}{T_2} = \beta^* \times U_H \Rightarrow b_{1,2} = T_1 \times \left(\beta^* \times U_H - \frac{e_{2,1}}{T_2} \right) = 3$$

Similarly, when $J_{1,2}$ completes after executing for $e_{1,2} = 1$ time unit, $b_{2,2}$ will again be updated depending on $\max\{e_{1,1}, e_{1,2}\}$. To ensure that the total allocation is proportional to $\beta^* \times U_H$, $b_{2,2}$ is updated as follows.

$$b_{2,2} = T_2 \times \left(\beta^* \times U_H - \frac{\max\{e_{1,1}, e_{1,2}\}}{T_1} \right) = 2$$

243 Since $J_{2,2}$ has an execution requirement as much as $E_{2,2} = 3(> b_{2,2})$ time units,
 244 a mode-switch to HC mode is triggered at $t = 13$.

□

245 Thus, at all times we need to store for each HC task τ_i the largest execution
 246 time of any job of that task in the busy interval. We reserve a budget proportional
 247 to this largest execution time for all future jobs of that task.

248 For each HC task τ_i , initialize $e_i^m = 0$, where e_i^m denotes the maximum amount
 249 of time for which any job of τ_i has executed in the **latest busy interval**. Let $b_{i,j}$
 250 denote the budget allocated to the j th job of τ_i . The MEBA budget allocation
 251 strategy comprises four functions which are shown in Algorithm 1

- 252 – Function COMPUTE_BI computes the LC budget $b_{i,j}$ for $J_{i,j}$.
- 253 – Function TRIGGER is used to switch the system mode from LC to HC.
- 254 – Function UPDATE_E is used to record the value of e_i^m .
- 255 – Function RESET_E is used to reset the value of e_i^m .

Now we present the situations in which each function in MEBA should be executed. RESET_E would be executed if the system experiences an idle instant. Figure 3 shows the situations in which Function COMPUTE_BI, TRIGGER and UPDATE_E should be executed. Suppose HC job $J_{i,j}$ is dispatched to the processor by the scheduler, then function COMPUTE_BI will be executed before $J_{i,j}$ starts to execute. Function TRIGGER will be executed if HC job $J_{i,j}$ has consumed $b_{i,j}$ units of LC budget in total but does not complete. Function UPDATE_E would be executed when HC job $J_{i,j}$ gets preempted or completes at the current time instant.

Algorithm 1 MEBA

```

1: function COMPUTE_BI      ▷ This function is called when HC job  $J_{i,j}$  is allocated to the
   processor at the current time instant.
2:   if System Mode=LC then
3:     Before  $J_{i,j}$  starts to execute, update LC budget


$$b_{i,j} = T_i \times \left( \beta^* \times U_H - \sum_{\tau_k \in \tau_H \setminus \tau_i} \frac{e_k^m}{T_k} \right)$$


4:   end if
5: end function
6:
7: function TRIGGER ▷ This function is called when HC job  $J_{i,j}$  has consumed  $b_{i,j}$  units of
   LC budget in total but does not complete.
8:   Switch System Mode to HC.
9: end function
10:
11: function UPDATE_E ▷ This function is called if HC job  $J_{i,j}$  gets preempted or completes
   at the current time instant.
12:   Update

$$e_i^m = \max \{e_i^m, e_{i,j}\}$$

   where  $e_{i,j}$  denote the total budget consumed by  $J_{i,j}$  so far.
13: end function
14:
15: function RESET_E      ▷ This function is called if the system experiences an idle instant.
16:   Reset  $e_i^m = 0$  for each HC task  $\tau_i$ .
17: end function

```

The following lemma records an important property of MEBA.

Lemma 1 (Mode-switch condition) *Suppose a mode-switch to HC mode is triggered at some time instant t^* in a **busy interval** under MEBA. For each HC task τ_i and each time instant $t \leq t^*$, let $e_{i,t}^m$ denote the maximum amount of time for which any job of τ_i has executed before t in that busy interval. Then, for all $t \leq t^*$,*

$$\sum_{\tau_i \in \tau_H} \frac{e_{i,t}^m}{T_i} \leq \beta^* \times U_H. \quad (1)$$

Further, t^ is the earliest time instant in that busy interval such that there is a job J_{k,h^*} of some HC task τ_k with execution requirement $E_{k,h^*} > e_{k,t^*}^m$, and*

$$\sum_{\tau_i \in \tau_H} \frac{e_{i,t^*}^m}{T_i} = \beta^* \times U_H. \quad (2)$$

Fig. 3 Situations to call functions in MEBA

273

274 *Proof* Since a busy interval only has one mode-switch to HC mode under MEBA,
 275 t^* is the only such mode-switch instant in the busy interval under consideration.

Suppose Equation (1) does not hold, and t denotes the earliest time instant where it fails. Let J_{k,h^*} denote a job of HC task τ_k that was continuously scheduled in the interval $[t', t)$ for some $t' < t$. The value of b_{k,h^*} was updated at t' to $T_k \times (\beta^* \times U_H - \sum_{\tau_i \in \tau_H \setminus \tau_k} e_{i,t'}^m / T_i)$ and remains the same up to t . Further, since t is the earliest time instant at which Equation (1) fails, we know that $e_{k,t'}^m \leq T_k \times (\beta^* \times U_H - \sum_{\tau_i \in \tau_H \setminus \tau_k} e_{i,t'}^m / T_i) = b_{k,h^*}$. Further, $\forall \tau_i \in \tau_H \setminus \tau_k, e_{i,t}^m = e_{i,t'}^m$, because none of these tasks executed in the interval $[t', t)$. Then, since Equation (1) fails at t , we get,

$$\begin{aligned}
 e_{k,t}^m &> T_k \left(\beta^* \times U_H - \sum_{\tau_i \in \tau_H \setminus \tau_k} e_{i,t}^m / T_i \right) \iff \\
 e_{k,t}^m &> T_k \left(\beta^* \times U_H - \sum_{\tau_i \in \tau_H \setminus \tau_k} e_{i,t'}^m / T_i \right) \iff e_{k,t}^m > b_{k,h^*}
 \end{aligned}$$

276 Since $e_{k,t'}^m \leq b_{k,h^*} < e_{k,t}^m$ and J_{k,h^*} is the only job executing in $[t', t)$, it
 277 must be the case that J_{k,h^*} itself has executed for a total time of $e_{k,t}^m$ by t . This,
 278 combined with the fact that J_{k,h^*} did not execute for more than $e_{k,t'}^m$ before t' ,
 279 indicates that there is some time instant in $[t', t)$ when J_{k,h^*} has completed b_{k,h^*}
 280 units of execution but remains incomplete. By definition of MEBA, this would
 281 have triggered a mode-switch to HC mode at that time instant. This is impossible
 282 however, because $t^* (\geq t)$ is the only mode-switch in the busy interval. Thus we
 283 have shown that Equation (1) holds for all $t \leq t^*$.

Since a mode-switch is triggered at t^* , from the definition of MEBA we know that there is a job J_{k,h^*} of some HC task τ_k that remains incomplete at t^* . By t^* , J_{k,h^*} has executed for exactly $\lambda_{k,t^*} = b_{k,h^*} = T_k \times (\beta^* \times U_H - \sum_{\tau_i \in \tau_H \setminus \tau_k} e_{i,t^*}^m / T_i)$ time units. Therefore at t^* , we have $\sum_{\tau_i \in \tau_H \setminus \tau_k} e_{i,t^*}^m / T_i + \lambda_{k,t^*} / T_k = \beta^* \times U_H$, and the total execution demand E_{k,h^*} of J_{k,h^*} is greater than λ_{k,t^*} . Then $e_{k,t^*}^m = \lambda_{k,t^*}$, because Equation (1) holds at t^* and $e_{k,t^*}^m \geq \lambda_{k,t^*}$ by definition. Thus we have identified the conditions that must hold when a mode-switch is triggered under

MEBA. Since t^* is the only mode-switch in this busy interval, we can conclude that t^* is the earliest time instant when these conditions are satisfied. $\square$

Finally, the following lemma shows that for a given β^* , MEBA is optimal in terms of its ability to postpone the mode-switch when compared to any other offline or runtime strategy that uses a fixed budget for each HC task in the LC mode.

Lemma 2 *For the dynamic MC task system τ , consider any budget allocation $B_i^L (\forall \tau_i \in \tau_H)$ satisfying the condition $\sum_{\tau_i \in \tau_H} B_i^L / T_i \leq \beta^* \times U_H$, such that each job of each HC task τ_i is given a budget of exactly B_i^L in the LC mode. Then, for any job sequence (release time and execution demand) of τ , the mode-switch instant based on this budget allocation is no later than the mode-switch instant under MEBA.*

Proof We prove this lemma by contradiction. Suppose under MEBA the mode-switch happens at t^* for some job sequence. From Lemma 1 we know that

$$\sum_{\tau_i \in \tau_H} e_{i,t^*}^m / T_i = \beta^* \times U_H$$

, where e_{i,t^*}^m denotes the maximum amount of time for which any job of τ_i has executed before t^* in the present busy interval. Also, there is at least one HC task τ_k having an incomplete job J_{k,h^*} at t^* with execution demand $E_{k,h^*} > e_{k,t^*}^m$.

Suppose there exists a budget allocation satisfying the condition

$$\sum_{\tau_i \in \tau_H} B_i^L / T_i \leq \beta^* \times U_H$$

that can further postpone the mode-switch. Then it must be the case that $\forall \tau_i \in \tau_H : B_i^L \geq e_{i,t^*}^m$ and $B_k^L > e_{k,t^*}^m$. However this is impossible because the assumption $\sum_{\tau_i \in \tau_H} B_i^L / T_i \leq \beta^* \times U_H$ is violated in that case. $\square$

5 MEBA with Budget Reclamation (MEBA+)

An important advantage of the dynamic model is that it can reduce the probability that a system switches to HC mode when compared to the static model. However, the mode-switch condition in Equation 1, i.e.,

$$\sum_{\tau_i \in \tau_H} \frac{e_i^m}{T_i} \leq \beta^* \times U_H$$

is more likely to be violated when the system experiences a long busy interval because the value of e_i^m can only be reset to 0 after an idle instant. Therefore, in this section, we propose a budget reclamation scheme to further prevent the occurrence of mode-switch.

Previous works e.g. [27] usually exploit the unused budget of hard real-time tasks to serve soft real-time tasks, and soft real-time tasks would run under the control of a pseudo hard real-time task. These works aim at utilizing the unused budget to serve soft real-time tasks in addition to the hard real-time tasks. In

contrast, the budget reclamation scheme in this paper aims at preventing the occurrence of mode-switch.

Suppose a job $J_{i,j}$ has consumed $e_{i,j}$ units of LC budget by time instant t , the $e_{i,j}$ units of budget can be classified into two categories:

1. **Definition 2 (Unused Budget)** Among the $e_{i,j}$ units of budget, there are $\phi_{i,j}$ units of unused budget (U-budget) from jobs with higher priority.
2. **Definition 3 (Own Budget)** Among the $e_{i,j}$ units of budget, there are $o_{i,j}$ units of its own budget (O-budget), where $o_{i,j} = e_{i,j} - \phi_{i,j}$.

The technique to calculate the value of $o_{i,j}$ will be presented in Section 5.1, and here we assume that the value of $o_{i,j}$ is known.

With the budget reclamation scheme, (i.e. in MEBA+), we will use the O-budget ($o_{i,j}$) instead of total budget ($e_{i,j}$) to control mode-switch. That is, a HC job will trigger a mode-switch only when it has consumed $b_{i,j}$ units of its O-budget (i.e., $o_{i,j} = b_{i,j}$), but does not signal completion. Hence, the function used to control mode-switch in MEBA+, TRIGGER+, is modified as shown in Algorithm 2.

Algorithm 2 TRIGGER+

- 1: **function** TRIGGER+ ▷ This function is called when HC job $J_{i,j}$ has consumed $b_{i,j}$ units of O-budget without completion.
 - 2: Switch System Mode to HC.
 - 3: **end function**
-

Let o_i^m denote the maximum amount of O-budget that any job of τ_i has consumed in the **latest busy interval**. Then function COMPUTE_BI+ in MEBA+ would use o_i^m instead of e_i^m to determine $b_{i,j}$ for $J_{i,j}$. The details of COMPUTE_BI+ is shown in Algorithm 3.

Algorithm 3 COMPUTE_BI+

- 1: **function** COMPUTE_BI+
- 2: **if** System Mode=LC **then**
- 3: Before $J_{i,j}$ starts to execute at the current time instant, update LC budget

$$b_{i,j} = T_i \times \left(\beta^* \times U_H - \sum_{\tau_k \in \tau_H \setminus \tau_i} \frac{o_k^m}{T_k} \right) \quad (3)$$

- 4: **end if**
 - 5: **end function**
-

In MEBA+, function UPDATE_E+ will update the value of $o_i^m = \max\{o_i^m, o_{i,j}\}$ if HC job $J_{i,j}$ gets preempted or completes at the current time instant. Similarly, function RESET_E+ in MEBA+ will reset the value of o_i^m to 0 once an idle instant occurs. The details of UPDATE_E+ and RESET_E+ are shown in Algorithm 4.

We have already introduced the way that MEBA+ determines the LC budget ($b_{i,j}$) of HC jobs and controls the mode-switch according to the O-budget ($o_{i,j}$) consumed by each job. In the following section, we present techniques to calculate the value of $o_{i,j}$.

Algorithm 4 UPDATE_E+ and RESET_E+

```

1: function UPDATE_E+ ▷ This function is called if HC job  $J_{i,j}$  gets preempted or
   completes at the current time instant.
2:   Update
   
$$o_i^m = \max\{o_i^m, o_{i,j}\}$$

3: 

---


4: end function
5: function RESET_E+ ▷ This function is called if the system experiences an idle instant.
6:   Reset  $o_i^m = 0$  for each HC task  $\tau_i$ .
7: end function

```

5.1 Functions to refill and reclaim U-budget

In this section, we present the techniques to calculate the value of $o_{i,j}$. We first define a budget queue (BQ) which is used to store the U-budget of completed jobs.

Definition 4 (BQ) Budget queue (BQ) is a queue of tuples $\chi = (g, d)$, where g denotes the amount of the U-budget, and d is the deadline which is also the expiration time of this budget. BQ is sorted by the deadline of tuples $(\chi.d)$ in ascending order.

5.1.1 Budget Refill

Function REFILL is shown in Algorithm 5, and it would be executed only if some job $J_{i,j}$ completes at the current time instant. Let $d_{i,j}^1 = r_{i,j} + D_i^L$ denote the absolute virtual deadline and $d_{i,j}^2 = r_{i,j} + T_i$ denote the absolute actual deadline for job $J_{i,j}$.

For a LC job, its LC budget B_i^L is fixed, but its LC budget has two priorities under EDF-UVD. First, if $o_{i,j} < \alpha_i \times B_i^L$, then $\alpha_i \times B_i^L - o_{i,j}$ units of budget has virtual deadline $d_{i,j}^1$, while the remaining $(1 - \alpha_i) \times B_i^L$ units of budget has deadline $d_{i,j}^2$. As shown in Algorithm 5, $(\alpha_i \times B_i^L - o_{i,j}, d_{i,j}^1)$ and $((1 - \alpha_i) \times B_i^L, d_{i,j}^2)$ will be inserted into BQ in this case. Otherwise if $o_{i,j} > \alpha_i \times B_i^L$, then the $B_i^L - o_{i,j}$ units of budget only has deadline $d_{i,j}^2$, and hence $(B_i^L - o_{i,j}, d_{i,j}^2)$ will be inserted into BQ.

On the other hand, a HC job does not have a fixed LC budget. Suppose HC job $J_{i,j}$ completes at the current time instant. Since at least o_i^m units of LC budget is reserved for future jobs of τ_i . If $o_{i,j} < o_i^m$, a tuple $(o_i^m - o_{i,j}, d_{i,j}^1)$ is inserted into BQ.

5.1.2 Budget reclaim

The underlying tenet of the budget reclamation step is that, the budget in BQ from jobs with highest priority (earliest deadline) will always be consumed first. However, we should note that the U-budget of $\chi \in BQ$ would be expired by its deadline $\chi.d$. Before job $J_{i,j}$ starts to execute, $o_{i,j}$ would be initialized to 0. As $J_{i,j}$ executes from t_s to t_e **continuously**, $o_{i,j}$ would also increase correspondingly (i.e., $o_{i,j} = o_{i,j} + (t_e - t_s)$) under the control of the execution time monitor. Suppose function RECLAIM in Algorithm 6 is called at t_e and it has identified

Algorithm 5 REFILL($J_{i,j}, BQ$)

```

1: function REFILL( $J_{i,j}, BQ$ ) ▷ This function is called if  $J_{i,j}$  completes at the current time
   instant.
2:   if  $J_{i,j}$  completes at the current time instant then
3:     if  $L_i = LC$  and  $o_{i,j} < B_i^L$  then
4:       if  $o_{i,j} < B_i^L \times \alpha_i$  then
5:         Insert  $(\alpha_i \times B_i^L - o_{i,j}, d_{i,j}^1)$  into BQ.
6:         Insert  $((1 - \alpha_i) \times B_i^L, d_{i,j}^2)$  into BQ.
7:       else
8:         Insert  $(B_i^L - o_{i,j}, d_{i,j}^2)$  into BQ.
9:       end if
10:    end if
11:    if  $L_i = HC$  and  $o_{i,j} < o_i^m$  then
12:      Insert  $(o_i^m - o_{i,j}, d_{i,j}^1)$  into BQ.
13:    end if
14:  end if
15: end function

```

371 that there are $\phi_{i,j}$ units of U-budget among the $t_e - t_s$ units of budget that $J_{i,j}$
 372 has consumed. Then $o_{i,j}$ is updated as follows.

$$o_{i,j} = o_{i,j} - \phi_{i,j} \quad (4)$$

373
 374 Now we introduce how to calculate the amount of U-budget ($\phi_{i,j}$) among the
 375 $t_e - t_s$ units of total budget consumed by $J_{i,j}$ during $[t_s, t_e]$. In RECLAIM, $\phi_{i,j}$ is
 376 initialized to 0, and then we loop through BQ to search for eligible tuple $\chi \in BQ$
 377 with earlier deadline than $d_{i,j}$.

1. If $\chi.g + \phi_{i,j} < t_e - t_s$, then all the U-budget of χ would be consumed by $J_{i,j}$.
 In this case, we update

$$\phi_{i,j} = \phi_{i,j} + \chi.g$$

378 and remove χ from BQ. Then we go to the next iteration.
 379 2. Else if $\chi.g + \phi_{i,j} \geq t_e - t_s$, then at most $t_e - t_s - \phi_{i,j}$ units of U-budget of χ
 380 can be consumed by $J_{i,j}$ by t_e . Thus, we update $\chi.g = \chi.g - (t_e - t_s - \phi_{i,j})$ and
 381 $\phi_{i,j} = t_e - t_s$. After this, we would jump out of the loop, and return $o_{i,j} - \phi_{i,j}$
 382 which is the amount of O-budget consumed by $J_{i,j}$ by t_e .

383 5.2 Execution Flows of Functions in MEBA+

384 In this Section, we present the situations in which functions in MEBA+ should be
 385 executed. RESET_E is executed only if the system experiences an idle instant. In
 386 Figure 4, we show the execution flow of function COMPUTE_BI+, TRIGGER+,
 387 UPDATE_E+, REFILL and RECLAIM.

388 Once a new job $J_{i,j}$ is released, it is waiting for execution in the ready queue.
 389 Suppose $J_{i,j}$ is dispatched to the processor by the scheduler and $L_i = HC$, then
 390 COMPUTE_BI+ is executed before it starts to execute. Suppose $J_{i,j}$ starts to

continues to execute. Otherwise, TRIGGER+ will be executed and the system switches to HC mode.

Similar to Lemma 1, the following lemma presents the mode-switch condition under MEBA+ and EDF-UVD.

Lemma 3 (Mode-switch condition) *Suppose a mode-switch to HC mode is triggered at some time instant t^* in a **busy interval** under MEBA+. For each HC task τ_i and each time instant $t \leq t^*$, let $o_{i,t}^m$ denote the maximum amount of O-budget that any job of τ_i has consumed before t in that busy interval. Then, for all $t \leq t^*$,*

$$\sum_{\tau_i \in \tau_H} \frac{o_{i,t}^m}{T_i} \leq \beta^* \times U_H. \quad (5)$$

Further, t^* is the earliest time instant in that busy interval such that there is a job J_{k,h^*} of some HC task τ_k has $o_{k,h^*} = o_{k,t^*}^m$ but does not signal completion, and

$$\sum_{\tau_i \in \tau_H} \frac{o_{i,t^*}^m}{T_i} = \beta^* \times U_H. \quad (6)$$

Proof Similar to Lemma 1. □

5.3 Analyze the Complexity of EDF-UVD and MEBA+

For the static model consisting of n tasks, EDF-VD [28] can be implemented efficiently with a runtime complexity of $O(\log n)$ per event, where an event is either the arrival of a job (event 1), completion of a job (event 2) or the occurrence of mode-switch (event 3).

The priority queue data structure proposed in [29] supports the operations of inserting (insert) and deleting the smallest item (deleteMin) or a specified item (deleteSpec) in logarithmic time, and the operation of finding the smallest item (min) in constant time. We maintain two such priority queues, Q_L and Q_H in the implementation of EDF-UVD and MEBA+.

Suppose the system is in LC mode initially, and $J_{x,y}$ denotes the currently executing job. We summarize the operations that MEBA+ with EDF-UVD perform to deal with different events as follows.

1. A new job $J_{i,j}$ arrives at t .
 - (a) Insert $J_{i,j}$ into Q_L and Q_H , prioritized according to virtual deadline D_i^L and actual deadline T_i , respectively in $O(\log n)$ time.
 - (b) Suppose $J_{x,y}$ is no longer the minimum job in Q_L and gets preempted by $J_{i,j}$ (only $J_{i,j}$ could have earlier deadline than $J_{x,y}$). Then RECLAIM ($O(n)$) and UPDATE_E+ (only for HC job) ($O(1)$) is executed for $J_{x,y}$ in sequence,
 - (c) If $J_{i,j}$ is a HC job, before $J_{i,j}$ starts to execute, COMPUTE_BI+ ($O(1)$) is executed,
2. The currently executing job $J_{x,y}$ completes at t .
 - (a) Then RECLAIM ($O(n)$), UPDATE_E+ (only for HC job) ($O(1)$) and RE-FILL ($O(1)$) is executed for $J_{x,y}$ in sequence.
 - (b) Delete $J_{x,y}$ from Q_L (deleteMin) and Q_H (deleteSpec) in $O(\log n)$ time.

- (c) Suppose $J_{i,j}$ is the minimum job in Q_L and $J_{i,j}$ is a HC job, then COMPUTE.BI+ ($O(1)$) is executed.
3. The currently executing LC job $J_{x,y}$ has executed for α_i time units.
- (a) RECLAIM ($O(n)$) is executed for $J_{x,y}$. Delete $J_{x,y}$ from Q_L (deleteMin) and insert $J_{x,y}$ back to Q_L according to its actual deadline T_i in $O(\log n)$ time.
4. The currently executing job $J_{x,y}$ triggers the potential mode-switch condition.
- (a) RECLAIM ($O(n)$) is executed for $J_{x,y}$.
- (b) If the value of $o_{x,y}$ is reduced by RECLAIM, the $J_{x,y}$ continues to execute. Otherwise, the scheduler switches to the priority queue Q_H .

As we can observe from the above description, function RECLAIM has a runtime complexity $O(n)$. As a result, MEBA+ with EDF-UVd can be implemented with $O(n)$ runtime complexity per event in the worst time. However each LC job would generate one additional event when compared to EDF-VD [3]. However, MEBA with EDF-UVd does not have RECLAIM and REFILL operations and hence it has a runtime complexity $O(\log n)$

6 Schedulability Test

As introduced in Section 3.1, each LC task in the dynamic model can receive $\alpha_i \times C_i$ units of budget per job when the system is in HC mode. Thus, α_i denotes the proportion of utilization that is statically reserved for LC task τ_i in HC mode.

Let α^* denote the service level of all the LC tasks (τ^L) in HC mode which is equal to the proportion of U_L that is statically reserved for all the LC tasks in HC mode. Given a task set τ characterized by the dynamic model, α^* can be computed as follows.

$$\alpha^* = \frac{\sum_{\tau_i \in \tau^L} \alpha_i \times u_i}{U_L} \quad (7)$$

In this section, we derive a sufficient schedulability test for MEBA with EDF-UVd given the values of α^* and β^* .² Then, we show that any system that is schedulable by MEBA with EDF-UVd would also be schedulable by MEBA+ with EDF-UVd.

Theorem 1 *Given α^* and β^* , a dynamic MC task system τ is MC-schedulable by MEBA and EDF-UVd on an uniprocessor platform if the following condition holds:*

$$(1 - \alpha^*)(1 - \beta^*) \geq \frac{U_H + U_L - 1}{U_L \times U_H}, \quad (8)$$

and the virtual deadline factor x falls in the following range:

$$\left[\frac{\beta^* \times U_H + \alpha^* \times U_L}{1 - U_L \times (1 - \alpha^*)}, \frac{1 - U_H - \alpha^* \times U_L}{(1 - \alpha^*) \times U_L} \right]$$

Proof We prove this theorem by mapping the dynamic MC task system τ scheduled by MEBA and EDF-UVd to a static MC task system τ' scheduled by EDF-VD. Consider a job sequence $\mathcal{J}$ (set of release times and execution demand) of τ that

² Techniques for determining α^* and β^* are presented in Section 7.

results in a mode-switch at some time instant t^* . Let e_{i,t^*}^m denote the maximum amount of time for which any job of HC task $\tau_i \in \tau$ has executed up to instant t^* in the busy interval containing t^* .

Consider the mapping of tasks from the dynamic model to the static model shown in Table 2. For a LC task $\tau_i = (T_i, C_i, LC)$ with virtual deadline $D_i(L) = T_i \times x$, we map it to a HC task $\tau'_{i,1}$ and a LC $\tau'_{i,2}$. For a HC task $\tau_i = (T_i, C_i, HC)$ with virtual deadline $D_i(L) = T_i \times x$, we map it to a HC task τ'_i .

Task	T_i	C_i^L	C_i	L_i	$D_i(L)$
$\tau'_{i,1}$	T_i	$\alpha_i \times C_i$	$\alpha_i \times C_i$	HC	$T_i \times x$
$\tau'_{i,2}$	T_i	-	$(1 - \alpha_i) \times C_i$	LC	-
τ'_i	T_i	e_{i,t^*}^m	C_i	HC	$T_i \times x$

Table 2 Tasks in static MC system τ'

Let $\tau' = \{\tau'_{i,1}, \tau'_{i,2} | \forall \tau_i \in \tau_L\} \cup \{\tau'_i | \forall \tau_i \in \tau_H\}$. In Lemma 4, we would show that for the job sequence $\mathcal{J}$ generated by the dynamic system τ , it is feasible to generate an *identical* job sequence $\mathcal{J}'$ by the static system τ' . Since the runtime scheduling policy of EDF-UVd and EDF-VD are also identical, if τ' is MC-Schedulable under EDF-VD, then job sequence $\mathcal{J}$ of τ is also guaranteed to be MC-Schedulable under MEBA and EDF-UVd. Thus, we can use the schedulability test for EDF-VD [3] to derive a test for MEBA and EDF-UVd.

From Equation 7, we have $\sum_{\tau_i \in \tau_L} \alpha_i \times \frac{C_i}{T_i} = U_L \times \alpha^*$, and hence the total utilization of LC tasks in τ' (tasks of type $\tau'_{i,2}$ in Table 2) is $\sum_{\tau_i \in \tau_L} \frac{(1 - \alpha_i) \times C_i}{T_i} = U_L \times (1 - \alpha^*)$.

The total utilization of HC tasks in τ' (tasks of type $\tau'_{i,1}$ and τ'_i in Table 2) in the HC mode is

$$\sum_{\tau_i \in \tau_H} \frac{C_i}{T_i} + \sum_{\tau_i \in \tau_L} \frac{\alpha_i \times C_i}{T_i} = U_H + U_L \times \alpha^*.$$

From Lemma 1 we have $\sum_{\tau_i \in \tau_H} \frac{e_{i,t^*}^m}{T_i} \leq U_H \times \beta^*$. Thus the total utilization of HC tasks in τ' in the LC mode is

$$\sum_{\tau_i \in \tau_H} \frac{e_{i,t^*}^m}{T_i} + \sum_{\tau_i \in \tau_L} \alpha_i \times \frac{C_i}{T_i} = U_H \times \beta^* + U_L \times \alpha^*.$$

Then, using Theorems 1 and 2 from [3], we get that τ' is MC-Schedulable by EDF-VD on an uniprocessor platform with virtual deadline factor x , if

$$x \leq \frac{1 - U_H - \alpha^* \times U_L}{(1 - \alpha^*) \times U_L} \wedge x \geq \frac{\beta^* \times U_H + \alpha^* \times U_L}{1 - U_L \times (1 - \alpha^*)}$$

and $(1 - \alpha^*) (1 - \beta^*) \geq \frac{U_H + U_L - 1}{U_L \times U_H}.$

This proves the theorem because above equations are independent of any parameters specific to $\mathcal{J}$. $\square$

Lemma 4 For each job sequence $\mathcal{J}$ generated by the dynamic system τ in Theorem 1, it is feasible to generate an identical job sequence $\mathcal{J}'$ in the static system τ' given by Table 2. Here identical means that for each job $J \in \mathcal{J}$, there is a job $J' \in \mathcal{J}'$ such that J and J' have the same release time, execution demand and virtual and original deadlines.

Proof First we consider some simplifications to jobs of $\mathcal{J}$. For any $J \in \mathcal{J}$ released after t^* such that J is a job of a LC task τ_i , we assume that its execution demand is bounded by $\alpha_i \times C_i$. This is reasonable because the job cannot execute beyond this bound under EDF-UVD. Also, for any $J \in \mathcal{J}$ released at t such that J is a job of a LC task τ_i , if J executes for more than $\alpha_i \times C_i$ time units (say E_i), then we replace it with two jobs J_a and J_b . J_a is released at t , has a demand of $\alpha_i \times C_i$ and deadline at $t + xT_i$. J_b is also released at t , has a demand of $E_i - \alpha_i \times C_i$ and deadline at T_i . Note that under EDF-UVD, J_b will begin execution only after J_a completes, and hence their combined schedule is identical to that of J .

Consider the following job sequence in the static model (denoted as $\mathcal{J}'$). For each job $J \in \mathcal{J}$ released at time instant t , having execution demand E and deadline at t' , a job J' will be released by τ' with the same parameters, such that:

1. If J is a job of HC task τ_i , then static HC task τ'_i will release the job J' .
2. If J is a job of LC task τ_i , $t < t^*$ and J has deadline at $t + T_i$ (job type J_b), then static HC task $\tau'_{i,2}$ will release the job J' .
3. If J is a job of LC task τ_i , and $t \geq t^*$ (job in the HC mode) or J has deadline at $t + xT_i$ (job type J_a), then static HC task $\tau'_{i,1}$ will release the job J' .

We now show that $\mathcal{J}'$ is valid, that is it is feasible to generate such a sequence. The periods of the mapped tasks in the static model are identical to the periods of the corresponding tasks in the dynamic model. Also, for each job execution demand E in the dynamic model, the corresponding execution demands in the static model are no larger than the task execution time parameters. For case 1 it is easy because $E \leq C_i$. For case 2, $E \leq (1 - \alpha_i)C_i$ because it is of type J_b . Finally, for case 3, $E \leq \alpha_i \times C_i$ because either it is of type J_a or it is a job released after t^* . Similarly, it can be seen from the mapping that the deadlines can also be matched as long as mode-switch is triggered in the static model at t^* .

Now we show that the mode-switch is indeed triggered at t^* even in the static model. Observe that the runtime scheduling policy of EDF-UVD and EDF-VD are identical (they both use EDF). Hence, up to time instant t^* , since the job sequences in $\mathcal{J}$ and $\mathcal{J}'$ are identical, the schedule is equivalent (whenever a job J is scheduled in the dynamic model, corresponding job J' is scheduled in the static model). This means, at t^* , if a job $J \in \mathcal{J}$ has remaining execution time, then job $J' \in \mathcal{J}'$ also has the same amount of remaining execution. From Lemma 1 we know that at t^* there is an incomplete job J of a HC task $\tau_i \in \tau$ with $E > e_{i,t^*}^m$, where E denotes the total execution requirement of J . Then, we can conclude that J' of HC task $\tau'_i \in \tau'$ also has remaining execution $E' = E > e_{i,t^*}^m$ at t^* . Further, by definition we know that for each HC task $\tau_i \in \tau$, no job has executed for more than e_{i,t^*}^m time units before t^* . Then, t^* is the first time instant at which any job in $\mathcal{J}'$ is requesting for more execution than its LC budget (since $\mathcal{J}$ and $\mathcal{J}'$ have equivalent schedule until t^*). This concludes the proof. $\square$

Finally, we prove that any system that is MC-schedulable by MEBA and EDF-UVD would also be MC-schedulable by MEBA+ and EDF-UVD. We first use an example to explain the intuition behind the proof for this claim.

528 **Definition 5 (Job Instance)** A job instance of jobs is a set of jobs $\mathcal{J} = \{J_{1,1}, \dots\}$
 529 with a collection of execution requirements $\{E_{1,1}, \dots\}$.

Fig. 5 Execution Sequence with Budget Reclamation

Fig. 6 Execution Sequence without Budget Reclamation

Example 3 Suppose τ is schedulable by MEBA and EDF-UVD, but is not MC-schedulable by MEBA+ and EDF-UVD. As shown in Figure 5, S is the execution

sequence of job instance $\mathcal{J} = \{J_{1,1}, J_{2,1}, J_{2,2}, J_{3,1}\}$ of τ when it is scheduled by MEBA+ and EDF-UVD, where $\beta^* = 0.6$ and the execution requirements of these jobs are

$$E_{1,1} = 2, E_{2,1} = 2, E_{2,2} = 2, E_{3,1} = 5$$

530 In S , $J_{3,1}$ misses its deadline at $t = 10$.

If we update the execution requirements as follows

$$\hat{E}_{1,1} \leftarrow E_{1,1} + 1 = 3, \hat{E}_{2,2} \leftarrow E_{2,2} + 1 - 1 = 2, \hat{E}_{3,1} \leftarrow E_{3,1} - 1 = 4$$

531 , then we can get a new job set $\hat{\mathcal{J}}$ of τ .

532 As shown in Figure 6, $\hat{S}$ is the execution sequence of $\hat{\mathcal{J}}$ when it is scheduled by
533 MEBA+ and EDF-UVD. As we can observe, no budget reclamation happens in
534 $\hat{S}$, and hence $\hat{S}$ is also the execution sequence of $\hat{\mathcal{J}}$ when it is scheduled by MEBA
535 and EDF-UVD. In $\hat{S}$, the deadline miss still happens at $t = 10$.

536 By creating such a contradiction, we can prove that any system that is MC-
537 schedulable by MEBA and EDF-UVD, would also be MC-schedulable by MEBA+
538 and EDF-UVD. □

539 Let S denote the execution sequence of a job instance $\mathcal{J}$ when it is scheduled
540 by MEBA+ and EDF-UVD during a busy interval $[0, t_f]$. Suppose the earliest
541 budget reclamation happens between $J_{x,y}$ and $J_{a,b}$ in S during $[t_s^1, t_e^1]$, and the
542 budget consumed by $J_{a,b}$ during $[t_s^1, t_e^1]$ is assumed to be the U-budget of $J_{x,y}$. For
543 example, in Figure 5, $J_{2,2}$ has consumed the U-budget of $J_{1,1}$ during $[3, 4]$. The
544 following lemma records an important property of S .

545 **Lemma 5** If $\hat{E}_{x,y} \leftarrow E_{x,y} + t_e^1 - t_s^1$ and $\hat{E}_{a,b} \leftarrow E_{a,b} - (t_e^1 - t_s^1)$, then $J_{x,y} \mapsto \hat{J}_{x,y}$,
546 $J_{a,b} \mapsto \hat{J}_{a,b}$ and $\mathcal{J} \mapsto \hat{\mathcal{J}}$ (Symbol $A \mapsto B$ denotes mapping A to B). Let $\hat{S}$ denote
547 the execution sequence of $\hat{\mathcal{J}}$ when it is scheduled by MEBA+ and EDF-UVD, then
548 $\hat{S}$ has the following property:

- 549 1. The system still stays in LC mode by t_e^1 in $\hat{S}$.
- 550 2. The mapped jobs $\hat{J}_{a,b}$ and $\hat{J}_{x,y}$ have $\hat{o}_{a,b} = o_{a,b}$, $\hat{e}_{a,b} = e_{a,b} - (t_e^1 - t_s^1)$, $\hat{o}_{x,y} =$
551 $o_{x,y} + (t_e^1 - t_s^1)$ and $\hat{e}_{x,y} = e_{x,y} + (t_e^1 - t_s^1)$.

552 *Proof* In S , the earliest budget reclamation firstly happens between $J_{x,y}$ and $J_{a,b}$,
553 and the budget consumed by $J_{a,b}$ during $[t_s^1, t_e^1]$ is assumed to be the U-budget of
554 $J_{x,y}$. Then we can infer two facts:

- 555 1. There no U-budget from jobs with higher priority than $J_{x,y}$ during $[t_s^1, t_e^1]$.
- 556 2. $J_{a,b}$ is the active job (i.e., a job that does not complete) with the highest
557 priority during $[t_s^1, t_e^1]$.

558 Suppose $J_{x,y}$ completes at $t_y (< t_s^1)$ in S , then any job executed during $[t_y, t_s^1]$
559 must have higher priority than $J_{x,y}$ because otherwise the first budget reclamation
560 should occur before t_s^1 .

561 Thus, in $\hat{S}$, $\hat{J}_{x,y}$ still has an execution requirement of $t_e^1 - t_s^1$ at t_s^1 . As a result,
562 $\hat{J}_{x,y}$ is the active job with highest priority during $[t_s^1, t_e^1]$ in $\hat{S}$ as long as the system
563 stays in LC mode during $[t_s^1, t_e^1]$. This implies that if a mode-switch is triggered
564 during $[t_s^1, t_e^1]$, it could only be triggered by $\hat{J}_{x,y}$.

Since $J_{x,y}$ has at least $t_e^1 - t_s^1$ units of U-budget at t_s^1 , then we have $o_{x,t_s^1}^m - o_{x,y} \geq t_e^1 - t_s^1$. Suppose the mode-switch is triggered at $t^* \in (t_s^1, t_e^1)$ in $\hat{S}$, then at t^* , $\hat{J}_{x,y}$ has (from Lemma 3)

$$\left(\hat{o}_{x,y} = \hat{o}_{x,t^*}^m = o_{x,y} + (t^* - t_s^1)\right) \wedge \left(o_{x,t_s^1}^m \geq o_{x,y} + t_e^1 - t_s^1\right) \Rightarrow \hat{o}_{x,t^*}^m < o_{x,t_s^1}^m$$

However, this contradicts the fact that $\hat{o}_{x,t^*}^m \geq \hat{o}_{x,t_s^1}^m = o_{x,t_s^1}^m$.

Thus, there is no mode-switch in $\hat{S}$ during $[t_s^1, t_e^1]$, and hence $\hat{J}_{x,y}$ will execute during $[t_s^1, t_e^1]$ using its O-budget to meet the increased execution requirement. Thus, in $\hat{S}$, no budget reclamation happens during $[t_s^1, t_e^1]$, and we have $\hat{o}_{a,b} = o_{a,b}$, $\hat{e}_{a,b} = e_{a,b} - (t_e^1 - t_s^1)$, $\hat{o}_{x,y} = o_{x,y} + (t_e^1 - t_s^1)$ and $\hat{e}_{x,y} = e_{x,y} + (t_e^1 - t_s^1)$. $\square$

Based on Lemma 5, we present the following theorem and the proof for it.

Theorem 2 Suppose a MC system τ is MC-schedulable by MEBA and EDF-UVD, then it would still be MC-schedulable by MEBA+ and EDF-UVD.

Proof Suppose τ is schedulable by MEBA and EDF-UVD, but is not MC-schedulable by MEBA+ and EDF-UVD. Let S denote an execution sequence of a job set $\mathcal{J}$ scheduled MEBA+ and EDF-UVD during a busy interval $[0, t_f]$, where job $J_{h,k}$ firstly misses its deadline at t_f .

Suppose the k_{th} budget reclamation happens between $J_{x,y}$ and $J_{a,b}$ in S during $[t_s^k, t_e^k]$. Let's apply Lemma 5 recursively, and update $\hat{E}_{x,y} \leftarrow E_{x,y} + t_e^k - t_s^k$, $\hat{E}_{a,b} \leftarrow E_{a,b} - (t_e^k - t_s^k)$ for $k = 1, 2, 3, \dots$ one by one. Then we can get new job instances $\hat{\mathcal{J}}^1, \hat{\mathcal{J}}^2, \dots, \hat{\mathcal{J}}^n$. Let $\hat{S}^k$ denote execution sequence of $\hat{\mathcal{J}}^k$ when it is scheduled by MEBA+ and EDF-UVD, then after k iterations, no budget reclamation happens by t_e^k .

Suppose the last reclamation in S happens during $[t_s^n, t_e^n]$, then no budget reclamation happens in $\hat{S}^n$, and each job has

$$\hat{E}_{i,j} = E_{i,j} - \phi_{i,j} + \varphi_{i,j}$$

$$\hat{e}_{i,j} = \hat{o}_{i,j} = e_{i,j} - \phi_{i,j} + \varphi_{i,j}$$

where $\phi_{i,j}$ denotes the amount of U-budget $J_{i,j}$ has consumed and $\varphi_{i,j}$ denotes the amount of $J_{i,j}$'s U-budget that are consumed by other jobs in S . Thus, $\hat{S}^n$ is also the execution sequence of $\hat{\mathcal{J}}^n$ when it is scheduled by MEBA and EDF-UVD.

Since $\hat{\mathcal{J}}^n$ is by schedulable by MEBA and EDF-UVD, then no deadline miss happens in $\hat{S}^n$. Thus, $\hat{J}_{h,k}$ must meet its deadline and hence we have $\hat{E}_{h,k} = \hat{o}_{h,k} = \hat{e}_{h,k} \Rightarrow E_{h,k} = e_{h,k}$. However, this contradicts the condition $E_{h,k} > e_{h,k}$ because $J_{h,k}$ misses its deadline at t_f in S . Therefore the assumption is false and the lemma is true. $\square$

7 Determination of α^* and β^*

From Theorem 1 in Section 6 we know that a dynamic system τ is MC-Schedulable under MEBA and EDF-UVD if $(1 - \alpha^*)(1 - \beta^*) \geq M$ where $M = \frac{U_H + U_L - 1}{U_L \times U_H}$.

We can see that there is a trade-off between β^* and α^* . A higher value for β^* implies a higher total budget allocation for HC tasks in the LC mode, and consequently the mode-switch can get delayed. It is then possible that LC tasks would continue to receive a higher service level ($= 1$) in the LC mode for a longer duration of time. On the other hand, a higher value for α^* implies higher service level for LC tasks in the HC mode, but then the resulting smaller value for β^* may lead to a mode-switch at an earlier time instant. Thus, β^* controls the amount of time for which LC tasks receive full service, whereas α^* controls the minimum guaranteed service for LC tasks at all times.

If we want to minimize the likelihood that the system switches to HC mode, then we can set $\alpha^* = 0$ and choose the maximum possible value for $\beta^* (= 1 - M)$. Alternatively, if we wish to support LC tasks as much as possible, while ensuring that the likelihood of mode-switch is relatively small, then we can set $\beta^* = \frac{\sum_{\tau_i \in \tau_H} C_i^L / T_i}{U_H}$, where C_i^L denotes the maximum observed execution time for HC task τ_i in large-scale simulations.

Note that the value of β^* is upper bounded by $1 - M$. Therefore, if τ has a large $U_H + U_L$ value (small $1 - M$) and $1 - M \ll \frac{\sum_{\tau_i \in \tau_H} C_i^L / T_i}{U_H}$, then the overall demand of HC tasks will frequently exceed $(1 - M)U_H$. As a result, the system is likely to switch to HC mode frequently. To support LC tasks as much as possible in this case, it might be better to set $\beta^* = 0$ and $\alpha^* = 1 - M$.

HC budget allocation for LC tasks. Given α^* , we have the flexibility to assign different HC budgets for each LC task τ_i , $B_i^H (= \alpha_i \times C_i)$, as long as $\sum_{\tau_i \in \tau_L} \frac{B_i^H}{T_i} = \sum_{\tau_i \in \tau_L} \frac{\alpha_i \times C_i}{T_i} \leq \alpha^* \times U_L$. A simple strategy in which we do not differentiate between LC tasks is to distribute the budget equally, i.e., $\forall \tau_i, \tau_j \in \tau_L : B_i^H = \frac{T_i}{T_j} B_j^H$. Another possible solution is that the application designer can provide a range of HC budget values for the LC tasks, and the system designer can choose among them based on α^* and the requirements of other LC tasks.

7.1 Total System Utilization

When the system is in the LC mode, each LC task τ_i receives a budget of C_i and all HC tasks combined receive a total budget proportional to $\beta^* \times U_H$. Thus the system utilization in the LC mode can be defined as follows.

$$SU_L = \beta^* \times U_H + \sum_{\tau_i \in \tau_L} \frac{C_i}{T_i} = \beta^* \times U_H + U_L$$

Similarly, when the system is in the HC mode, LC tasks receive a budget of $B_i^H = \alpha_i \times C_i$ and HC tasks receive a budget of C_i per job. Thus the system utilization in the HC mode can be defined as follows.

$$SU_H = \sum_{\tau_i \in \tau_L} \frac{B_i^H}{T_i} + U_H = \alpha^* \times U_L + U_H$$

Fig. 7 β^* and weight w when $SU(w)$ is maximized

Consider the following definition of *total system utilization*, where $w \in [0, 1]$

$$\begin{aligned}
 SU(w) &= w \times SU_L + (1 - w) \times SU_H \\
 &\leq U_H(1 - w + w\beta^*) + U_L(w + (1 - w)(1 - \frac{M}{1 - \beta^*})) \\
 &= U_H(1 - w) + U_L + U_H w\beta^* - \frac{U_L(1 - w)M}{1 - \beta^*}
 \end{aligned} \tag{9}$$

$SU(w)$ gives a value for the total system utilization, assuming a weight of w for the LC mode utilization. Since $SU(w)$ is a convex function of β^* and β^* is in the range $[0, 1 - M]$, by taking the first derivate and equating it to 0, $SU(w)$ is maximized when

$$\beta^* = \max \left\{ 0, \min \left\{ 1 - M, 1 - \sqrt{\frac{M(1 - w)U_L}{wU_H}} \right\} \right\} \tag{10}$$

We performed some experiments to understand the impact of weight w on the total system utilization. In Figures 7 we show how the value of β^* varies when $SU(w)$ is maximized. We fix $U_{sum} = U_H + U_L = 1.5$ and assign different values to U_L and w . Z-axis plots the value of β^* for which $SU(w)$ is maximized, x-axis denotes U_L , and y-axis denotes weight w . $U_L \in \{U_{sum} - 1, U_{sum} - 0.9, U_{sum} - 0.8, \dots, 1\}$ and $w \in \{0.02, 0.04, 0.06, \dots, 1\}$ in the figure.

As we can observe, in many cases, the value of β^* for which $SU(w)$ is maximized is not sensitive to the value of w . The reason for this observation is that when the system has a large U_{sum} , the value of $1 - M$ is relatively small. For such systems, $SU(w)$ is maximized when either $\beta^* = 1 - M$ or $\beta^* = 0$ for almost all the values of w .

8 Evaluation

An important advantage of the dynamic model is that the designer is not required to specify LC budgets for individual HC tasks. There are two other MC models that do not require the designer to provide such budgets. In the worst-case reservations model commonly used in the safety-critical industry today, tasks are always guaranteed to receive C_i units of budget per job, irrespective of their criticality level. The dynamic model generalizes this model, and they are equivalent if we set $\beta^* = 1$. Although this model ensures isolation between tasks at different criticality levels, it does not allow for efficient sharing of the processor. Another one is the elastic model [13], in which all HC tasks receive C_i units of budget per job, and the service level of LC tasks is fixed at $(1 - U_H)/U_L$. The dynamic model also generalizes this model, and they are equivalent if we set $\beta^* = 0$ and $\alpha^* = (1 - U_H)/U_L$. In the remainder of this section, we evaluate the dynamic model from three different aspects.

8.1 Minimum guaranteed service for LC tasks

In this section we experimentally evaluate the performance of the dynamic model in terms of its ability to provide minimum guaranteed service to LC tasks. We compare its performance to the service adaption strategy [16], which also uses a EDF-based scheduling policy and provides a minimum guaranteed service to LC tasks. This strategy uses EDF-VD and decreases the dispatch frequency of LC tasks when the system switches mode. It finds the minimal possible factor y (single value) to extend the periods of all LC tasks i.e., $T_i \leftarrow y \times T_i$, where $y \geq 1$. Therefore, the minimum guaranteed service for LC tasks that the service adaption strategy can support is equal to $\frac{1}{y}$. We compare this value with α^* in the dynamic model to evaluate their relative performance.

For this comparison, we have to use task systems characterized by the static model (i.e., given C_i^L value for each HC task τ_i), because the service adaptation strategy is designed for such systems. To ensure a fair comparison, we assume that β^* in the dynamic model is set to a value such that $\beta^* \times U_H = \sum_{\tau_i \in \tau_H} C_i^L / T_i$. We then compute the maximum possible α^* that still guarantees schedulability.

We use the same task set generation procedure as in [8], and it can be summarized as follows. Each task is generated based on the following parameters.

- Task τ_i is a HC task with probability $PH = 0.5$.
- C_i^L is drawn using an uniform distribution over $[1, 10]$.
- C_i is drawn using an uniform distribution over $[C_i^L, RC \times C_i^L]$, where $RC \in \{3, 4, 5\}$.
- T_i is drawn using an uniform distribution over $[C_i, 200]$.

Task set τ is empty initially. Randomly generated tasks based on the above procedure are added to the task set repeatedly. Let $U_A = \frac{U_L + U_H + U_H \beta^*}{2}$ denote the average utilization of task set τ in LC and HC mode at any point in the generation process (computed based on tasks that have already been added to τ). We classify τ based on the range in which U_A lies: $[0.54, 0.55]$, $[0.59, 0.60]$, $[0.64, 0.65]$, $[0.69, 0.7]$, $[0.74, 0.75]$. We do not consider the case when $U_A > 0.75$, because very few task sets with $U_A > 0.75$ are schedulable under the service adaption strategy. A new

task is added to τ until U_A falls in the range we choose. However if U_A becomes greater than the upper bound of the range, we discard the entire task set and repeat the process.

Table 3 Average minimum guaranteed service for LC tasks

Average Utilization	0.55	0.6	0.65	0.7	0.75
Dynamic Model RC=3	0.985	0.931	0.832	0.566	0.235
Service Adaption RC=3	0.976	0.882	0.636	0.268	0.177
Dynamic Model RC=4	0.988	0.950	0.831	0.643	0.321
Service Adaption RC=4	0.984	0.903	0.639	0.326	0.129
Dynamic Model RC=5	0.978	0.912	0.648	0.295	0.089
Service Adaption RC=5	0.964	0.805	0.339	0.210	0.053

In Table 3, we show the average minimum guaranteed service for LC tasks that the dynamic model and service adaption strategy can support when $RC = 3, 4$ and 5 . Each value in the table is based on 1000 task sets. If a task set is not schedulable by either the dynamic model or the service adaptation strategy, then the service level of LC tasks is assumed to be 0. This is reasonable because there is no guarantee on deadlines for such task sets. Even though both the service adaption strategy and the dynamic model are based on EDF, the dynamic model always outperforms the service adaption strategy. For some settings, its performance is almost two times better than the performance of the latter.

The reasons resulting in this performance gap can be summarized as follows: 1) the schedulability analysis for service adaption strategy is based on pessimistic approximate demand bound functions whereas for the dynamic model it is based on utilization-based tests, and 2) we set virtual deadlines for LC tasks to reduce the scenario where jobs of LC tasks receive unnecessary budget (advantage of EDF-UVD over EDF-VD). Another important advantage of the dynamic model is that it provides the flexibility to set different HC service levels for each LC task (α^* can be split among the different LC tasks based on α_i), while the service adaption strategy has to decrease the dispatch frequency of all the LC tasks to the same degree.

8.2 Analysis of probability of mode-switch

Under the dynamic model the mode-switch is triggered only when the total allocated LC budget for all HC tasks combined exceeds $\beta^* \times U_H$. This is different from the static model in which a mode-switch is triggered even when a single HC task exceeds its allocated budget. Therefore, the dynamic model can reduce the probability that a task system switches mode when compared to the static model. In this section we analytically compare the two models from this aspect. We first use a simple example to illustrate how the dynamic model can reduce the likelihood of mode-switch.

Example 4 Suppose we have a task system $\tau = \{\tau_1, \tau_2, \tau_3\}$ as given in Table 4.

Using Equation 8, we set $\beta^* = 0.25$ and $\alpha^* = 0$. Thus, as long as $(e_1^m/10 + e_2^m/10)/0.8 \leq 0.25$, i.e., $e_1^m + e_2^m \leq 2$, the system would stay in the LC mode

Table 4 Example Task Set

Task	T_i	C_i	L_i
τ_1	10	5	LC
τ_2	10	4	HC
τ_3	10	4	HC

(from Lemma 1). Instead, if we have a fixed budget as in the static model, e.g., $B_1^L = B_2^L = 1$, the system would switch to HC mode when either τ_2 or τ_3 executes beyond 1 time unit. Thus, in the dynamic model, even if τ_2 executes for 1.05 time units, the system could stay in the LC mode as long as no job of τ_3 has executed beyond 0.95 time units.

Let $P(s_i)$ denote the probability that no job of τ_i executes for more than $C_i \times s_i$ time units in a certain busy interval, where s_i is a random variable in the range $(0, 1]$. Then, a task set τ does not switch mode under the static model only when no HC task executes beyond its fixed LC budget $B_i^L = C_i^L$ determined offline (let $C_i^L = \beta_i \times C_i$ for each i). Then, assuming all tasks are independent, the probability that the system does not switch to HC mode is

$$P_{noswitch}^s = \prod_{\tau_i \in \tau_H} P(s_i = \beta_i) \quad (11)$$

On the other hand, according to Lemma 1, the system does not switch mode under the dynamic model as long as $\sum_{\tau_i \in \tau_H} \frac{e_{i,t}^m}{T_i} \leq \beta^* \times U_H$. Suppose $e_{i,t}^m = s_i \times C_i$. Then, the probability that the system does not switch to HC mode under the dynamic model is

$$P_{noswitch}^d = \sum_{\substack{s_i \in (0,1] \\ \sum_{\tau_i \in \tau_H} \frac{s_i \times C_i}{T_i} \leq \beta^* \times U_H}} \left(\prod_{\tau_i \in \tau_H} P(s_i) \right) \quad (12)$$

We can see that $P_{noswitch}^d \geq P_{noswitch}^s$, because $s_i = \beta_i$ is one possible assignment that satisfies $\sum_{\tau_i \in \tau_H} \frac{s_i \times C_i}{T_i} \leq \beta^* \times U_H$ (using Lemma 2), but there exists many more assignments of s_i that also satisfy the above condition. Further, we can also observe that the probability of mode-switch under the static model increases exponentially with the number of HC tasks, while the dynamic model can mitigate this problem.

Table 5 Example distribution for $P(s_i)$

s_i	0.1	0.2	0.3	0.4	0.5
$P(s_i)$	0.01	0.05	0.2	0.5	0.8
s_i	0.6	0.7	0.8	0.9	1.0
$P(s_i)$	0.9	0.95	0.98	0.995	1.0

From Equations 11 and 12, we can evaluate the performance of dynamic and static models analytically in terms of their ability to reduce the mode-switch probability. Now we plot Equation 11 and 12 using a specific distribution for $P(s_i)$. We

set $\beta^* = 1 - M = (1 - U_L)(1 - U_H)/(U_L \cdot U_H)$ and $\alpha^* = 0$ in the dynamic model because this choice postpones the mode-switch as much as possible. For the static model, to ensure a fair comparison, we set $\beta_i = 1 - M \Leftarrow C_i^L = (1 - M) \times C_i$ for each HC task τ_i . Thus, both the models have the same LC system service level for HC tasks (β^*). But the dynamic model uses the runtime strategy MEBA to distribute this service among the HC tasks, while the static model uses $\beta_i \times C_i$ as a fixed LC budget.

Fig. 8 Comparison of the probability that the mode-switch does not happen

Suppose τ comprises n HC tasks and $P(s_i)$ conforms to the distribution shown in Table 5 for each task. Then we can calculate $P_{noswitch}^s$ and $P_{noswitch}^d$ when $n \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. Figure 8 plots Equations 11 and 12 as a function of the number of HC tasks, where x-axis denotes the number of HC tasks and y-axis denotes the probabilities. The mode-switch probability depends on the value of $\beta^* = 1 - M$. Hence, we show the results for $1 - M \in \{0.45, 0.55, 0.65, 0.75\}$. When $1 - M > 0.75 \vee 1 - M < 0.45$, the probability would either approach 1 or 0 according to the distribution in Table 5. Hence we do not present the results for $1 - M > 0.75 \vee 1 - M < 0.45$.

We can see that when $1 - M$ has a large value, the mode-switch probability for both dynamic and static models is low, and vice versa. Also, the dynamic model can significantly reduce this probability when compared to the static model under different settings. We can observe that as the number of tasks increase, $P_{noswitch}^s$ decreases monotonically, while the value of $P_{noswitch}^d$ is stable.

Note that the mode-switch probability for the static model that we computed is in fact the probability of mode-switch in several MC studies (e.g., [5–7, 3, 8, 9,

16, 12, 19, 10, 11]). Other works (e.g., [15, 14, 18, 17]) have different levels of mode-switch depending on the number of HC tasks that are executing beyond their LC budgets; in these studies the first mode-switch has a probability identical to what we computed for the static model. Hence, the results of Figure 8 provide a direct comparison between the dynamic model and all the above studies.

8.3 Simulation results about mode-switch

In the previous section, we compare the probability of mode-switch in dynamic and static model analytically. In this section, we compare 1) MEBA+ (with EDF-UVD), 2) MEBA (with EDF-UVD) and 3) static LC budget allocation (with EDF-VD) through simulations.

Task Set Generator: We use the same task set generator as described in Section 8.1 to generate random MC task set characterized by the static model for simulation. We generate task sets with $U_A \in \{0.7, 0.8\}$ and $RC \in \{3, 5\}$. For each setting, we choose 5000 task sets that are schedulable by both EDF-VD and EDF-UVD for simulation. For fair comparison, in the simulation of MEBA+ or MEBA (with EDF-UVD), we set

$$\beta^* = \frac{1}{U_H} \sum_{\tau_i \in \tau_H} C_i^L / T_i.$$

Simulation Process: In the simulation, each task will release a job every T_i time units from time instant $t = 0$ to $t = 10000$. However, LC jobs are only released in LC mode and would be dropped once the system switches to HC mode. Jobs in the simulation are scheduled by 1) MEBA+ (with EDF-UVD), 2) MEBA (with EDF-UVD) and 3) static LC budget allocation (with EDF-VD).

At run-time, each job of τ_i will require $\lfloor s_i \times C_i \rfloor$ time units of execution. The probability that J_i requires $\lfloor s_i \times C_i \rfloor$ time units of execution is equal to $P(s_i)$, where $s_i \in \{0.1, 0.2, \dots, 1.0\}$ and $P(s_i)$ conforms to binomial distribution $B(10, 0.4)$ or $B(10, 0.6)$ as shown in Figure 9. To ensure a fair comparison, a list of execution requirements $[E_{i,1}, E_{i,2}, \dots]$ are generated off-line according to the distributions of $P(s_i)$. As a result, in the simulation of MEBA+ (with EDF-UVD), MEBA (with EDF-UVD) or static LC budget allocation (with EDF-VD), the j th job of τ_i would require the same amount ($E_{i,j}$) of execution.

Performance Metric: We focus on comparing 1) EDF-UVD with MEBA+, 2) EDF-UVD with MEBA and 3) EDF-VD (with static LC budget allocation) using the following two metrics.

1. The number of mode-switches occurred during $[0, 10000]$.
2. The percentage of time that the system stays in LC mode during $[0, 10000]$.

Figure 10 to 17 shows the simulation results of the three strategies using task sets with $U_A \in \{0.7, 0.8\}$, $RC \in \{3, 5\}$ and $P(s_i) \sim B(10, 0.4)$ or $B(10, 0.6)$. In these figures, the left y-axis denotes the percentage of time that the system stays in LC mode and the right y-axis denotes the number of mode-switches occurred in $[0, 10000]$. The x-axis of those figures denotes the number of HC tasks (n_H) in a task set.

Fig. 9 $P(s_i) \sim B(10, 0.4)$ and $P(s_i) \sim B(10, 0.6)$

As we can observe from the simulation results, the MEBA (or MEBA+) with EDF-UVD dominates the static LC budget allocation with EDF-VD under different settings. The performance gap between MEBA (or MEBA+) and static LC budget allocation also enlarges as n_H increases. This trend is in line with our analysis in Section 8.2. When the value of n_H reaches 8 or 9, very few mode-switches are triggered in the simulation of MEBA (or MEBA+) with EDF-UVD.

Besides, for MEBA, MEBA+ and static LC budget allocation, more mode-switches are triggered when $P(s_i) \sim B(10, 0.6)$ compared to the case when $P(s_i) \sim B(10, 0.4)$. When $P(s_i) \sim B(10, 0.6)$, each job is more likely to have a large execution requirement, and hence is more likely to trigger the mode-switch. As a result, the system also stays in LC mode for a longer time when $P(s_i) \sim B(10, 0.4)$.

Even though few mode-switches are occurred in the simulation of MEBA especially when the value of n_H is large. MEBA+ can further improve the performance of MEBA under different settings. For example, when $n_H = 9$, $P(s_i) \sim B(10, 0.4)$, $RC = 3$ and $U_A = 0.7$, there are more than 2 mode-switches on average occurred in the simulation of MEBA (with EDF-UVD). With budget reclamation scheme, MEBA+ can reduce the number of mode-switches to less than 1.

Fig. 10 $U_A = 0.7$, $RC = 3$ and $P(s_i) \sim B(10, 0.4)$

Fig. 11 $U_A = 0.7$, $RC = 5$ and $P(s_i) \sim B(10, 0.4)$

8.4 Total system utilization

We introduced a technique to maximize the total system utilization in Section 7. Here we compare the maximum total system utilization that dynamic and static models can support for different values of U_H and U_L . For the dynamic model this parameter can be calculated using Equation (10). For the static model, assuming it is scheduled by EDF-VD and no service guarantee for LC tasks after mode-switch, the total system utilization is maximized when $\sum_{\tau_i \in \tau_H} C_i^L / T_i = (1 - M)U_H$.

$$SU(w) = w.U_L + w.(1 - U_L)(1 - U_H)/U_L + (1 - w)U_H$$

Here w denotes the weight for LC mode. Since there is no service guarantee for LC tasks, the utilization is maximized by postponing the mode-switch as much as possible, i.e., similar to setting $\beta^* = 1 - M$ in the dynamic model.

Figure 18 and 19 shows the ratios between the total system utilization of dynamic and static models as a function of weight $w \in \{0.02, 0.04, \dots, 1.0\}$ when

Fig. 12 $U_A = 0.8$, $RC = 3$ and $P(s_i) \sim B(10, 0.4)$

Fig. 13 $U_A = 0.8$, $RC = 5$ and $P(s_i) \sim B(10, 0.4)$

$U_L + U_H = 1.1$ or 1.3 , respectively. As we can observe, when the weight for LC mode is relatively small, there is a clear performance gap between the two models. After the weight w exceeds a certain value, their performance overlaps. This indicates that when the weight w exceeds a certain value, the total system utilization is maximized when $\beta^* = 1 - M$. That is, if the weight for the LC mode is high enough and therefore we do not care about guaranteed service for LC tasks in the HC mode, then it is best to maximize the LC system service level for HC tasks (β^*).

9 Conclusion

In this paper we proposed a dynamic LC budget allocation mechanism for HC tasks to overcome the limitations of static execution estimates. Unlike the static model where the LC budget of each HC task is required to be provided by the

Fig. 14 $U_A = 0.7$, $RC = 3$ and $P(s_i) \sim B(10, 0.6)$

Fig. 15 $U_A = 0.7$, $RC = 5$ and $P(s_i) \sim B(10, 0.6)$

application designer, the dynamic model determines it at runtime based on observed job execution times. The system switches mode in the dynamic model only when the total LC budget allocation for all HC tasks combined is violated. We also proposed a mechanism that enables LC tasks to receive a minimum guaranteed budget allocation at all times, even in the HC mode. Then we presented metrics and explored the trade-off between the total LC budget allocation for HC tasks and the minimum guaranteed service for LC tasks. Finally, we also propose a budget reclamation scheme that can work along with the dynamic model and further prevent the occurrence of mode-switch.

In the future, we plan to further explore this trade-off between budget allocation for HC tasks and service guarantee for LC tasks. In particular, we plan to investigate techniques that can use the knowledge of task execution times (e.g., probabilistic worst-case execution time) to maximize the expected system utilization. Another direction of research is improving the runtime strategy for budget allocation to HC tasks. In MEBA we record the maximum execution time among

Fig. 16 $U_A = 0.8$, $RC = 3$ and $P(s_i) \sim B(10, 0.6)$

Fig. 17 $U_A = 0.8$, $RC = 5$ and $P(s_i) \sim B(10, 0.6)$

all the jobs in the past (i.e., e_i^m) to trigger a mode-switch. However it may be possible to further postpone the mode-switch by using other parameters (e.g., the sum of executions of all the jobs in the past), and we plan to explore such techniques.

References

1. P. J. Prisaznuk, "Integrated modular avionics," in *Aerospace and Electronics Conference (NAECON)*. IEEE, 1992, pp. 39–45.
2. S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*. IEEE, 2007, pp. 239–243.
3. S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, and A. Marchetti-Spaccamela, "The Preemptive Uniprocessor Scheduling of Mixed-Criticality Implicit-Deadline Sporadic Task Systems," in *ECRTS*, 2012.
4. A. Burns, "An augmented model for mixed criticality," in *In S. K. Baruah, L. Cucu-Grosjean, R. I. Davis, and C. Maiza, editors, Mixed Critical-ity on Multicore/Manycore*

Fig. 18 Total system utilization when $U_L + U_H = 1.1$

- Platforms (Dagstuhl Seminar 15121), volume 5. Schloss DagstuhlLeibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2015.
5. S. Baruah, A. Burns, and R. Davis, "Response-Time Analysis for Mixed Criticality Systems," in *RTSS*, 2011, pp. 34–43.
 6. S. Baruah and G. Fohler, "Certification-Cognizant Time-Triggered Scheduling of Mixed-Criticality Systems," *RTSS*, pp. 3–12, 2011.
 7. N. Guan, P. Ekberg, M. Stigge, and W. Yi, "Effective and Efficient Scheduling of Certifiable Mixed-Criticality Sporadic Task Systems," in *RTSS*, 2011, pp. 13–23.
 8. P. Ekberg and W. Yi, "Bounding and Shaping the Demand of Mixed-Criticality Sporadic Tasks," in *ECRTS*, 2012, pp. 135–144.
 9. A. Easwaran, "Demand-based Scheduling of Mixed-Criticality Sporadic Tasks on One Processor," in *RTSS*, 2013, pp. 78–87.
 10. A. Burns and S. Baruah, "Towards a More Practical Model for Mixed-Criticality Systems," in *Workshop on Mixed-Criticality Systems (co-located with RTSS)*, 2013.
 11. M. Jan, L. Zaourar, and M. Pitel, "Maximizing the execution rate of low-criticality tasks in mixed criticality system," in *Workshop on Mixed-Criticality Systems (co-located with RTSS)*, 2013.
 12. H. Su, N. Guan, and D. Zhu, "Service guarantee exploration for mixed-criticality systems," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, Aug 2014, pp. 1–10.
 13. H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '13. San Jose, CA, USA: EDA Consortium, 2013, pp. 147–152. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2485288.2485325>
 14. X. Gu, A. Easwaran, K.-M. Phan, and I. Shin, "Resource efficient isolation mechanisms in mixed-criticality scheduling," in *27th Euromicro Conference on Real-Time Systems*, July 2015, pp. 13–24.
 15. P. Huang, P. Kumar, N. Stoimenov, and L. Thiele, "Interference constraint graph-a new specification for mixed-criticality systems," in *Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on*, Sept 2013, pp. 1–8.
 16. P. Huang, G. Giannopoulou, N. Stoimenov, and L. Thiele, "Service adaptations for mixed-criticality systems," in *In Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

Fig. 19 Total system utilization when $U_L + U_H = 1.3$

17. T. Fleming and A. Burns, "Incorporating the notion of importance into mixed criticality systems," in *WMC*, 2014, p. 33.
18. J. Ren and L. T. X. Phan, "Mixed-criticality scheduling on multiprocessors using task grouping," in *27th Euromicro Conference on Real-Time Systems*. IEEE, 2015, pp. 25–34.
19. O. Gettings, S. Quinton, and R. I. Davis, "Mixed criticality systems with weakly-hard constraints," in *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, ser. RTNS '15. New York, NY, USA: ACM, 2015, pp. 237–246. [Online]. Available: <http://doi.acm.org/10.1145/2834848.2834850>
20. A. B. Sanjoy Baruah and Z. Guo, "Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors," in *28th Euromicro Conference on Real-Time Systems*, July 2016, pp. 131–138.
21. D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, and W. Yi, "Edf-vd scheduling of mixed-criticality systems with degraded quality guarantees," in *Real-Time Systems Symposium (RTSS), 2016 IEEE*. IEEE, 2016, pp. 35–46.
22. H. Su, N. Guan, and D. Zhu, "Service guarantee exploration for mixed-criticality systems," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*. IEEE, 2014, pp. 1–10.
23. F. Santy, L. George, P. Thierry, and J. Goossens, "Relaxing Mixed-Criticality Scheduling Strictness for Task Sets Scheduled with FP," in *ECRTS*, 2012, pp. 155–165.
24. F. Santy, G. Raravi, G. Nelissen, V. Nelis, P. Kumar, J. Goossens, and E. Tovar, "Two protocols to reduce the criticality level of multiprocessor mixed-criticality systems," in *Proceedings of the 21st International Conference on Real-Time Networks and Systems*, ser. RTNS '13. New York, NY, USA: ACM, 2013, pp. 183–192. [Online]. Available: <http://doi.acm.org/10.1145/2516821.2516834>
25. I. Bate, A. Burns, and R. I. Davis, "A bailout protocol for mixed criticality systems," in *27th Euromicro Conference on Real-Time Systems*, 2015, pp. 259–268.
26. Z. Guo, L. Santinelli, and K. Yang, "Edf schedulability analysis on mixed-criticality systems with permitted failure probability," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2015 IEEE 21st International Conference on*. IEEE, 2015, pp. 187–196.
27. R. I. Davis, K. W. Tindell, and A. Burns, "Scheduling slack time in fixed priority pre-emptive systems," in *Real-Time Systems Symposium, 1993., Proceedings*. IEEE, 1993, pp. 222–231.

-
- 947 28. S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and
948 L. Stougie, "Scheduling Real-Time Mixed-Criticality Jobs," *IEEE Transactions on Com-*
949 *puters*, vol. 61, no. 8, pp. 1140–1152, 2012.
- 950 29. T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.