MILP-based Deadline Assignment for End-to-End Flows in Distributed Real-Time Systems

Bo Peng
Department of Computer Science
Wayne State University
Detroit, Michigan, USA
et7889@wayne.edu

Nathan Fisher
Department of Computer Science
Wayne State University
Detroit, Michigan, USA
fishern@wayne.edu

Thidapat Chantem
Department of Electrical and Computer Engineering
Virginia Tech
Arlington, Virginia, USA
tchantem@vt.edu

ABSTRACT

End-to-end flows, which have a set of chainlike subtasks, are widely used in distributed real-time systems. For instance, multimedia and automative applications require that subtasks finish executing on a chain of processors before their end-to-end deadlines. The scheduling of such chained subtasks decides the schedulability of a distributed real-time system. Since the subtask priority assignment problem is NP-hard in general, most heuristics are presented to schedule end-to-end flows in two separate steps. The first step calculates intermediate relative deadlines for frames, and the second step makes scheduling decisions under EDF scheduling. Because the quality of the priority assignment of subtasks will directly affect the schedulability of the distributed systems, the two separate steps may cause pessimism in schedulability analysis. To reduce potential pessimism, we combine the two steps in our novel dGMF-PA (distributed generalized multiframe tasks with parameter adaption) model. We present an algorithm based on mixed-integer linear programming for optimally selecting frame relative deadlines in the dGMF-PA model. An approximation algorithm is also proposed to reduce computational running time. Our approximation algorithm has a tunable speed-up factor of $1+\epsilon$ where ϵ can be arbitrarily small, with respect to the exact schedulability test of dGMF-PA tasks under EDF scheduling. Extensive experiments have shown that our approximation algorithm (which is a sufficient schedulability test) can schedule at most 44 % more than the HOSPA algorithm.

Keywords

distributed generalized multiframe tasks with parameter adaption model; distributed real-time system scheduling; mixed-integer linear programming; approximation algorithms; EDF scheduling; end-to-end flows

1. INTRODUCTION

RTNS '16, October 19-21, 2016, Brest, France

DOI: http://dx.doi.org/10.1145/2997465.2997498

A job in the sporadic task model has an individual continuous unit of work. Sporadic tasks are independent except for resource contention. Such simple models are concise and able to represent many applications in uniprocessor systems, but not precise enough to represent complex tasks in distributed systems. In practice, a multimedia function [19] or a network service [16] usually consists of subtasks which may have precedence constraints. An end-to-end flow models a precedence graph as a chain in which a subtask becomes ready to execute when its preceding subtasks on the chain complete. In distributed real-time systems, subtasks of end-to-end flows can be (sometimes have to be) assigned to execute on different processors. For instance, in common video applications, data needs to be transformed from analog signals to digital signals. The digital signals are transmitted over the networks and transformed back to analog signals at the client side. These three steps have precedence constraints and can be modeled as an end-to-end flow.

The schedulability analysis for such distributed real-time systems is drawing increased attention, as real-time applications are becoming more and more complex. Since the problem of optimal task assignment in distributed real-time systems is NP-hard [15], we assume that subtasks are statically assigned to processors before a schedulability test is performed and focus instead on the priority assignment of subtasks. Due to the NP-hardness [17] of priority assignment for subtasks in end-to-end flows, many heuristics have been presented. The schedulability analysis of most heuristics consists of two independent steps. The first step is priority assignment and the second step utilizes the assignment to make scheduling decisions. A priority assignment such as PD (Proportional Deadline Algorithm) [17] can be efficient. PD assigns subtasks relative deadlines that are proportional to their execution times. However, such analysis often introduces pessimism as schedulability hinges upon the effectiveness of the priority assignment of subtasks. Iterative-based methods [26] have been considered to improve schedulability ratio. In such methods, the current iteration of priority assignment is calculated based on the parameters of the system in preceding iterations, and the assignment can affect the parameters in the next iterations. The iterations stop when the system is schedulable or some stopping criterion is met. However, pessimism also exists in iterative-based methods since the priority assignment and schedulability analysis of a system are not considered jointly in each iteration.

In order to reduce potential pessimism, we combine pri-

ority assignment under EDF scheduling and schedulability analysis together into one framework which utilizes mathematical programming. Two combined algorithms are developed under our dGMF-PA (distributed generalized multiframe tasks with parameter adaption) model which extends the GMF-PA (generalized multiframe tasks with parameter adaption) model [25]. The dGMF-PA model can represent end-to-end flows in distributed systems. The GMF-PA model (which extends the GMF model [5]) consists of a set of frames, each of which contains an execution time, a range of relative deadlines, and a range of minimum inter-arrival times. We refer to the minimum inter-arrival time among frames as period for simplicity. We refer to an end-to-end flow as a task and a subtask as a frame to be congruent with the dGMF-PA model. The insight of our work is that deadlines and periods of frames are flexibly chosen to increase the schedulability of distributed systems.

The first algorithm presented is a necessary schedulability test (in general) under EDF scheduling. The algorithm becomes an exact schedulability test and can select relative deadlines and periods of frames when parameters are integers. An approximation algorithm, which is proved to be (in general) a sufficient schedulability test, can reduce the running time and select its frame parameters. We also prove the speed-up factor is $1+\epsilon$ where ϵ can be arbitrarily small, with respect to the exact schedulability test of dGMF-PA tasks under EDF scheduling. Note that the two algorithms are both offline algorithms. In other words, parameters are fixed once parameter assignment and schedulability test have been completed.

Section 2 surveys the related work pertaining to our dGMF-PA model and the end-to-end flow model. We review the GMF model and introduce our dGMF-PA model in Section 3. Section 4 presents our combined algorithm which uses mixed-integer linear programming (MILP) to get a necessary schedulability test under EDF scheduling. An approximation algorithm of MILP is presented in Section 5. In Section 6, we conduct extensive experiments and compare them with state-of-the-art results. At last, Section 7 concludes this work and proposes future work.

2. RELATED WORK

In this section, we survey the related work pertaining to the dGMF-PA model in Section 2.1, and the related work of scheduling distributed end-to-end flows in Section 2.2.

2.1 The dGMF-PA Model

The dGMF-PA model is transformed from the generalized multiframe task (GMF) model (see details in Section 3.1). The GMF model was introduced by Baruah et al. [5] to extend the sporadic task model and multiframe task model (MF) [20]. Frames in the GMF model are also executed in order and thus form a "cycle" which can recur an infinite number of times. In the non-cyclic GMF task model [21], frames can execute out of order and thus reduce the pessimism of the modeling of software-defined radio [28]. The recurring real-time task (RRT) model [6] is a generalization of the GMF model to handle conditional codes. The non-cyclic recurring real-time task model [4] can generalize all the models referred to above.

The above models assume that parameters are fixed during task specification time. The GMF-PA model [25] relaxes this assumption by allowing parameters to be flexible and chosen under frame constraints and cycle constraints. In this paper, the dGMF-PA model is a distributed version of the GMF-PA model. Similar flexible models, such as the parameter-adaption model [10] and elastic model [9], are also used in many applications.

There are many applications based on the MF model and GMF model. Ding et al. [12] scheduled a set of tasks with the I/O blocking property under the MF model. Ekberg et al. [13] developed an optimal resource sharing protocol for the GMF model. Andersson [3] presented a schedulability analysis for the flows in multi-hop networks comprising software-implemented Ethernet switches, according to the GMF model. Peng and Fisher [25] presented a schedulability analysis for multiple-segment self-suspending tasks under EDF scheduling.

2.2 The Scheduling of End-to-End Flows

The schedulability analysis of distributed real-time systems has received much attention. Most applications in distributed real-time systems can be modeled by end-to-end flows/tasks/transactions in which subtasks/frames of a flow execute in a chainlike manner. Schedulability analysis of such applications has been proposed both for the fixed priority (FP) scheduling and earliest deadline first scheduling (EDF) algorithms.

For FP scheduling, Tindell and Clark [29] first proposed a holistic analysis, which was later improved by the offset-based analysis [22]. Such analysis calculates the worst-case response time of each subtask to set the offset and jitter of the succeeding subtask. The calculation is iterative. The FP scheduling of end-to-end flows was further improved by the offset based slanted technique [18] which exploits the interdependencies among subtasks using offsets.

For EDF scheduling, the offset-based analysis [23] was presented based on the similar analysis in FP scheduling. Pellizzoni and Lipari [24] provided new response time analysis and iterative algorithm to improve the schedulability analysis. In the iterative-based algorithms, deadline assignment affects the offsets and jitters of subtasks which in turn will affect the deadline assignment. Some existing deadline assignmnet algorithms include: PD [17], NPD [17], and HOSPA [26].

In this paper, we utilize the demand bound function [7] interface and mathematical programming to assign deadlines and analyze schedulability. The iterative-based algorithms cannot easily compute a demand bound function during an interval length because the response time of a subtask depends on the end-to-end flows in all processors.

3. MODEL

We review the GMF-PA model and define the dGMF-PA model in Section 3.1. We review the end-to-end flow model and apply our dGMF-PA model to end-to-end flows in Section 3.2.

3.1 The Distributed Generalized Multiframe Model with Parameter Adaption

In this section, we review the generalized multiframe tasks with parameter adaption (GMF-PA) model for uniprocessor systems and define the distributed generalized multiframe tasks with parameter adaption (dGMF-PA) model.

In the GMF-PA model [25], let $\mathcal{T} = \{\tau_0, \tau_1, ..., \tau_{n-1}\}$ be the task system consisting of n generalized multiframe

tasks executing on a uniprocessor system. Each task $\tau_i = [\phi_i^0, \phi_i^1, \phi_i^2, ..., \phi_i^{N_i-1}]$ consists of N_i frames. In each frame $\phi_i^j = (E_i^j, \underline{D}_i^j, \overline{D}_i^j, \underline{P}_i^j, \overline{P}_i^j)$, E_i^j is the execution time of the frame ϕ_i^j. The lower bound of the relative deadline D_i^j (minimum inter-arrival time between consecutive frames P_i^j) is $\underline{D}_i^j$ ($\underline{P}_i^j$), and the upper bound of D_i^j (P_i^j) is $\overline{D}_i^j$ ($\overline{P}_i^j$). That is, the parameters D_i^j and P_i^j are chosen in the ranges $[\underline{D}_i^j, \overline{D}_i^j]$ and $[\underline{P}_i^j, \overline{P}_i^j]$, respectively[1]. The N_i frames which execute in sequence can be seen as a cycle that executes infinitely. The cycle/end-to-end deadline [2] $\mathcal{D}_i$ is the upper bound of $D_i^{N_i-1} + \sum_{j=0}^{N_i-2} P_i^j$, and the period/cycle period $\mathcal{P}_i$ is the upper bound of $\sum_{j=0}^{N_i-1} P_i^j$. Figure 1 shows an example of a GMF-PA task when deadlines and periods of frames are assigned.

In order to generate proofs and our combined algorithms for distributed systems, we introduce our dGMF-PA model for distributed systems based on the GMF-PA model for uniprocessors. The dGMF-PA model can be reduced to the GMF-PA model when the number of processors is one. Let $\mathcal{T} = \{\tau_0, \tau_2, ..., \tau_{n-1}\}$ be the task system of n dGMF-PA tasks executing in a distributed system. Each task $\tau_i = [\tau_{i,0}, ..., \tau_{i,Q-1}]$ consists of Q virtual tasks on corresponding Q processors. Each virtual task $\tau_{i,p} = [\phi_{i,p}^0, \phi_{i,p}^1, \phi_{i,p}^2, ..., \phi_{i,p}^{N_i-1}]$ consists of N_i virtual frames and executes on processor p. Each virtual frame $\phi_{i,p}^j = (E_{i,p}^j, \underline{D}_{i,p}^j, \overline{D}_{i,p}^j, \underline{P}_{i,p}^j, \overline{P}_{i,p}^j)$ is similar to the frame in the GMF-PA model. In fact, there are only N_i frames in a GMF task τ_i and we require that each real frame must be statically assigned once on one processor. We call a virtual frame $\phi_{i,p}^k$ a real frame if the k'th frame of task τ_i is assigned on processor p, and we call a virtual frame $\phi_{i,p}^j$ an empty frame if the frame is not assigned on processor p. Figure 2 shows an example of the dGMF-PA model. In an empty frame $\phi_{i,p}^j$, we set $E_{i,p}^j = \underline{D}_{i,p}^j = \overline{D}_{i,p}^j = \underline{P}_{i,p}^j = \overline{P}_{i,p}^j = 0$. For simplicity, we also refer to a virtual frame as a frame (except when we specify a virtual frame as a real frame or an empty frame).

The cycle deadline $\mathcal{D}_i$ of task τ_i is the upper bound of $\sum_{p=0}^{Q-1} \left(D_{i,p}^{N_i-1} + \sum_{j=0}^{(N_i-2)} P_{i,p}^j \right)$, and the period/cycle period $\mathcal{P}_i$ is the upper bound of $\sum_{p=0}^{Q-1} \sum_{j=0}^{N_i-1} P_{i,p}^j$. The cycle deadline constraint intuitively represents the offset of the last frame's absolute deadline from the release time of the task and matches the traditional concept of an *end-to-end deadline*. The minimum execution time of the frame in τ_i is $E_i^{min} = \min\{\sum_{p=0}^{Q-1} E_{i,p}^0, \sum_{p=0}^{Q-1} E_{i,p}^1, ..., \sum_{p=0}^{Q-1} E_{i,p}^{N_i-1}\}$, and the total execution time of task τ_i is $E_i = \sum_{p=0}^{Q-1} \sum_{j=0}^{N_i-1} E_{i,p}^j$. The utilization of task τ_i is $U_{i,p} = \sum_{j=0}^{N_i-1} E_{i,p}^j / \mathcal{P}_i$, and the utilization of a task system is $U_p = \sum_{i=0}^{n-1} U_{i,p}$ on processor p. The maximum utilization of a processor in the distributed system is $U_{cap} = \max_{p=0}^{Q-1} U_p$.

In this paper, we consider that each frame of a task in the dGMF-PA model has its relative deadline constrained to be at most its period; that is, for all frames $\phi_{i,p}^j$, $D_{i,p}^j \leq P_{i,p}^j$. This assumption ensures that each frame has completed before the release time of the successive frame and simplifies the schedulability analysis for each processor.

We aim to optimally select relative deadlines ($D_{i,p}^j$) and make scheduling decisions in this paper, under the *basic requirements* as follows:

1. $E_{i,p}^j \leq \underline{D}_{i,p}^j \leq D_{i,p}^j \leq \overline{D}_{i,p}^j, \quad \forall i, j, p$
2. $E_{i,p}^j \leq \underline{P}_{i,p}^j \leq P_{i,p}^j \leq \overline{P}_{i,p}^j, \quad \forall i, j, p$
3. $D_{i,p}^j \leq P_{i,p}^j, \quad \forall i, j, p$

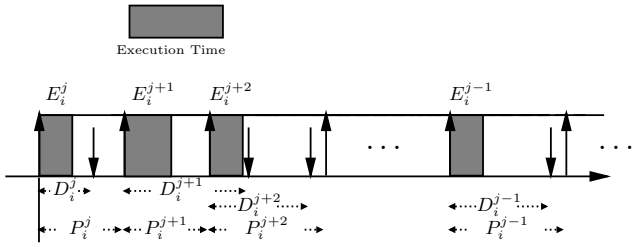


Figure 1: We omit " $\bmod N_i$" in all superscripts for simplicity, e.g., deadline D_i^{j+1} should be $D_i^{(j+1) \bmod N_i}$. This figure shows that N_i frames execute in sequence and release as soon as possible from the j'th frame to the $(j-1) \bmod N_i'$th frame.

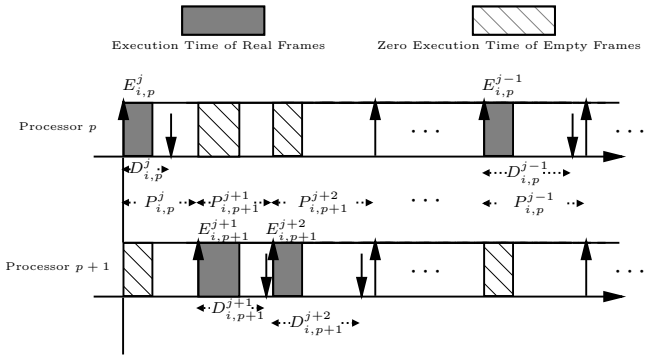


Figure 2: The dGMF-PA task τ_i contains two virtual tasks $\tau_{i,p}$ and $\tau_{i,p+1}$. $\phi_{i,p}^j$, $\phi_{i,p+1}^{j+1}$, $\phi_{i,p+1}^{j+2}$ and $\phi_{i,p}^{j-1}$ are real frames which execute in sequence (the real frames between $j+2$'th frame and $j-1$'th frame are omitted here).

[1]The specification of the bounds is out of the scope of this paper; instead, we set the bounds in a fairly general manner to compare our algorithm against existing work in Section 6.

[2]Note: the definition of $\mathcal{D}_i$ is not same as the one in the GMF-PA model [25]; we believe the new definition is more appropriate for modeling an end-to-end constraint.

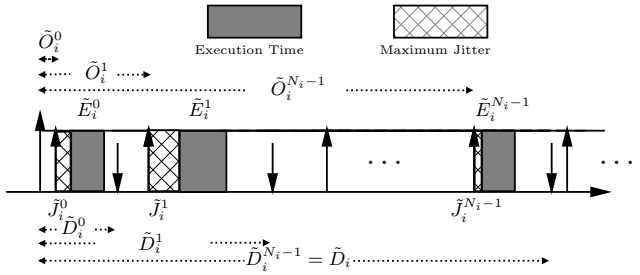


Figure 3: This end-to-end flow τ_i consists of N_i frames which can execute in different processors. The deadlines and jitters ensure the execution sequence of frames.

$$4.\ \sum_{p=0}^{Q-1}\left(D_{i,p}^{N_i-1}+\sum_{j=0}^{(N_i-2)}P_{i,p}^{j}\right)\le\mathcal{D}_i,\quad\forall i$$

$$5.\ \sum_{p=0}^{Q-1}\sum_{j=0}^{N_i-1}P_{i,p}^{j}\le\mathcal{P}_i,\quad\forall i$$

A task system must obey the first two inequalities to be feasible. The third inequality is the constrained frame deadline property. The last two inequalities check whether a system is feasible under the upper bounds $\mathcal{D}_i$ and $\mathcal{P}_i$. We refer to the first three constraints as *frame constraints* and the last two constraints as *cycle constraints* for the rest of this paper.

3.2 Distributed End-to-End Flows and the dGMF-PA Model

In this section, we review the distributed end-to-end flow model [24, 26] and apply the dGMF-PA model to the flows where each local processor is scheduled by EDF.

For distributed end-to-end flows, we will use a tilde over task parameters to distinguish from the dGMF-PA model. A task system $\widetilde{\mathcal{T}}=\{\widetilde{\tau}_0,\widetilde{\tau}_1,...,\widetilde{\tau}_{n-1}\}$ consists of n distributed end-to-end flows. Each task $\widetilde{\tau}_i=[\widetilde{\phi}_i^0,\widetilde{\phi}_i^1,\widetilde{\phi}_i^2,...,\widetilde{\phi}_i^{N_i-1}]$ consists of N_i real frames. In each frame $\widetilde{\phi}_i^j=(\widetilde{E}_i^j,\widetilde{D}_i^j,\widetilde{O}_i^j,\widetilde{J}_i^j)$, $\widetilde{E}_i^j$ is the execution time, $\widetilde{D}_i^j$ is the global relative deadline which is relative to the activation time of the task, $\widetilde{O}_i^j$ is the offset between the release time of a flow and the activation time of the frame $\widetilde{\phi}_i^j$, and $\widetilde{J}_i^j$ is the maximum jitter between the activation time and release time of the frame $\widetilde{\phi}_i^j$. The end-to-end deadline of the task $\widetilde{\tau}_i$ is $\widetilde{\mathcal{D}}_i$ and period between invocations of the task is $\widetilde{\mathcal{P}}_i$. Frames can execute on different processors and each frame can only be activated when its preceding frame completes executing. Figure 3 shows an example of an end-to-end flow model.

Now we translate a task in the end-to-end flow model to one in the dGMF-PA model. For each end-to-end frame $\widetilde{\phi}_i^j$, we create Q virtual dGMF frames $\phi_{i,p}^j$ for $p=0,1,\ldots,Q-1$. If the original frame $\widetilde{\phi}_i^j$ is assigned to processor p, all virtual frames $\phi_{i,q}^j$ where $q\neq p$ are empty frames in dGMF-PA. For a real frame $\phi_{i,p}^j$, the manner in which we set the parameter of the frame depends upon the setting: 1) if the offsets and global relative deadlines are not fixed by the designer, then we can set trivial lower and upper bounds to the frame period and relative deadline (i.e., $\underline{D}_{i,p}^j=\underline{P}_{i,p}^j=\widetilde{E}_i^j$ and $\overline{P}_{i,p}^j=\overline{D}_{i,p}^j=\widetilde{D}_i$); or 2) if the offsets and/or deadlines are fixed by the designer, then the trivial lower and upper bounds can be used again for the frame period and relative deadline; however, two additional constraints must be added: $\widetilde{O}_i^j=\sum_{q=0}^{Q-1}\sum_{\ell=0}^{j-1}P_{i,q}^{\ell}$ and $\widetilde{D}_i^j=D_{i,p}^j+\sum_{q=0}^{Q-1}\sum_{\ell=0}^{j-1}P_{i,q}^{\ell}$. Clearly, we can always set the frame execution $E_{i,p}^j$ to be $\widetilde{E}_{i,p}^j$. Jitter $\widetilde{J}_i^j$ can be modeled as a new independent dGMF-PA frame $\phi_{i,p}^{j'}$ in which $E_{i,p}^{j'}=0$ and $\underline{D}_{i,p}^{j'}=\overline{D}_{i,p}^{j'}=\underline{P}_{i,p}^{j'}=\overline{P}_{i,p}^{j'}=J_{i,p}^j$. This jitter frame is inserted before its corresponding frames (both empty and real) $\phi_{i,q}^j$ for all $q=0,\ldots,Q-1$; once the jitter frame $\phi_{i,p}^{j'}$ "completes", then the frame $\phi_{i,p}^j$ is ready to execute. The period of task τ_i is $\mathcal{P}_i=\widetilde{\mathcal{P}}_i$, and the end-to-end deadline is $\mathcal{D}_i=\widetilde{\mathcal{D}}_i$.

Due to the hardness of the frame assignment in distributed systems [15], we assume that real frames of end-to-end flows are statically assigned on processors (each jitter frame is bundled with its real frame on a processor). Aside from real frames on each processor p, we assign the other frames of all tasks to be empty frames on each processor p. That is, from the viewpoint of each processor p, all frames of tasks execute on processor p where some frames are empty.

In this paper, we deviate somewhat from the typical end-to-end flow semantics; in particular, in the end-to-end flow model, it is often assumed that a frame is released as soon as the previous frame signals it is complete. However, in this paper, we assume that a frame is eligible to execute only when its frame is released according to the period parameters of $P_{i,p}^j$. However, we conjecture that our schedulability results will continue to hold for the usual end-to-end semantics by permitting a frame to "early release" its job, but keeping its absolute deadline fixed to the same it would be in the dGMF-PA model (i.e., deadlines of frames do not shift when early released). The rationale is that fixing the deadlines but permitting early releases would only decrease the total execution demand and thus preserve schedulability. However, we leave proving this conjecture for future research.

Our combined parameter selection and schedulability test algorithms that are presented in Section 4 and 5 for the dGMF-PA model are thus applicable to distributed end-to-end flows.

4. THE EXACT DEADLINE ASSIGNMENT OF END-TO-END FLOWS IN THE DGMF-PA MODEL

In this section, we describe the combined technique of deadline selection for each frame and schedulability analysis under our dGMF-PA model by using mixed-integer linear programming (MILP) under EDF scheduling which utilizes demand and supply bound functions. The deadline of each frame is flexible subject to the frame and cycle constraints. Along with the selection, the schedulability analysis provides a necessary feasibility test for arbitrary real-valued task parameters. We prove the sufficiency and necessity of the schedulability test when task parameters are integers.

MILP is a mathematical optimization model that contains three parts: an objective function, constraint functions, and ranges of variables. A subset of variables can be restricted to integers in MILP. An MILP aims at finding the optimal value of the objective function under the restriction of constraint functions. We build our MILP to select the rela-

tive deadlines for all frames. At the same time, MILP gives a necessary feasibility test. However, note that for non-integer parameters, since the MILP is only necessary, the returned selection of deadlines may not be feasible. Later, in Section 5, we will give an approximation algorithm for the MILP that returns a feasible selection of relative deadlines for the non-integer case.

We first introduce the demand bound function and supply bound function which are used for schedulability analysis on a uniprocessor. We then show that our schedulability analysis for distributed systems breaks down to the analysis for uniprocessor systems. The demand bound function $\mathsf{dbf}_{i,t,p}$ accounts for the task τ_i's accumulated execution time of jobs which have both release time and deadline inside any interval of length t on processor p, and the supply bound function sbf_t gives the lower bound of resources that the system can supply over any interval of length t. Note that our MILP algorithm can consider different supply bound functions $\mathsf{sbf}_{t,p}$ for different processors. For simplicity, we consider the same supply bound functions sbf_t over all processors. In general, the sufficient and necessary condition for a uniprocessor feasible system is given in Equation 1.

$$\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p} \leq \mathsf{sbf}_t, \quad \forall t, p. \tag{1}$$

Since we allow deadlines to be variables and selected using MILP, the demand in this case is also treated as a variable. For instance, if we determine the demand over the interval length t from some frame $\phi_{i,p}^k$ that arrives at the beginning of the interval; if the relative deadline $D_{i,p}^k$ is set to be smaller or equal to t, then the demand from this job should be $E_{i,p}^k$; otherwise the demand is zero. Note that the demand of an empty frame $\phi_{i,p}^j$ is always zero since $E_{i,p}^j = 0$. Figure 4 illustrates a graph of this concept for an interval of length t. The detailed notations will be introduced later. Using the concepts above, Equation 1 becomes a set of constraint functions that a feasible system must obey to find a relative deadline assignment. In our algorithm, the supply bound function is $\mathsf{sbf}_t = t$ and the length t of any interval length is an integer. Our MILP can return an assignment if the system is schedulable. That is, we can determine the necessary feasibility of the system and select potential deadlines at the same time.

The general steps of our algorithm are as follows. For a given sequence of frames and a time interval of length t, we calculate the demand contribution of each frame to that interval length. Adding the demands of all frames generates the demand of a task, and adding the demands of all tasks (over all possible sequences of frames) generates the total demand in each processor at the time interval length t. The system is schedulable at a time interval length if the demand is no larger than the supply in all processors. We check all interval lengths, which are integers, in the algorithm.

For a given interval length t, we need to calculate the demand for every possible sequence of frames of task $\tau_{i,p}$ over any interval of length t and processor p. Assume that the first frame of $\tau_{i,p}$ to arrive in such an interval is $\phi_{i,p}^j$ (i.e., the j'th frame on processor p). The demand of any sequence starting with the j'th frame over a t-length interval is maximized if the j'th frame arrives exactly at the start of the interval and subsequent frames arrive as soon as possible (e.g., see Baruah et al. [5] for GMF schedulability). To calculate the demand from the k'th frame in such an interval for the specified sequence, $y_{i,t,p}^{j,k}$ represents the demand of this frame. We will calculate $y_{i,t,p}^{j,k}$ for all possible i, j, k, p, and t. For simplicity, we use "$\forall$" to represent the ranges of variables. The task index i ranges from 0 to $n-1$. The superscripts j and k represent the starting frame and the current frame respectively, and both have the ranges from 0 to $N_i - 1$. A processor p has the ranges from 0 to $Q-1$. It has been shown [5] that the maximum interval length is bounded by $O(\log n \cdot \frac{U_{cap}}{1-U_{cap}} \cdot \max_{\tau_i \in \tau}(\mathcal{P}_i - D_i^0))$ where $U_{cap} < 1$. We use $H = \lceil \log n \cdot \frac{U_{cap}}{1-U_{cap}} \cdot \max_{\tau_{i,p} \in \tau}(\mathcal{P}_i - E_i^{min}) \rceil$ as the maximum integer length interval since we do not know deadlines beforehand in our dGMF-PA model. Note that the range of any interval length t is shown in uniprocessor systems [5]. We choose the maximum utilization U_{cap} among processors to calculate the maximum interval length H. We use such abbreviations across this paper. The demand of the task $\tau_{i,p}$ starting from the j'th frame in time interval length t is $y_{i,t,p}^j$. The maximum demand of $\tau_{i,p}$ among all starting frames is $y_{i,t,p}$.

Deadline Assignment and Feasibility Test

Initialization: $E_{i,p}^k = D_{i,p}^k = P_{i,p}^k = 0$ if $\phi_{i,p}^k$ is an empty frame.

1 minimize: $\mathcal{L}$
2 subject to:
3 $\quad \boldsymbol{E_{i,p}^k \leq \underline{D}_{i,p}^k} \leq D_{i,p}^k \leq \boldsymbol{\overline{D}_{i,p}^k}, \quad \forall i, k, p.$
4 $\quad \boldsymbol{E_{i,p}^k \leq \underline{P}_{i,p}^k} \leq P_{i,p}^k \leq \boldsymbol{\overline{P}_{i,p}^k}, \quad \forall i, k, p.$
5 $\quad D_{i,p}^k \leq P_{i,p}^k, \quad \forall i, k, p.$
6 $\quad \sum_{p=0}^{Q-1} \sum_{k=0}^{N_i-1} P_{i,p}^k \leq \boldsymbol{\mathcal{P}_i}, \sum_{p=0}^{Q-1} \left(D_{i,p}^{N_i-1} + \sum_{j=0}^{(N_i-2)} P_{i,p}^j \right) \leq \boldsymbol{\mathcal{D}_i}, \forall i.$
7 $\quad y_{i,t,p}^{j,k} = x_{i,t,p}^{j,k} * \boldsymbol{E_{i,p}^k} + \lfloor \frac{\boldsymbol{t}}{\boldsymbol{\mathcal{P}_i}} \rfloor * \boldsymbol{E_{i,p}^k}, \quad \forall i, j, k, t, p.$
8 $\quad \frac{\boldsymbol{t}-t_b}{\boldsymbol{\mathcal{P}_i}} \leq x_{i,t,p}^{j,k} - \frac{\boldsymbol{realmin}}{\boldsymbol{\mathcal{P}_i}}, \quad \forall i, j, k, t, p.$
9 $\quad t_b = \left(\sum_{p=0}^{Q-1} \left(\sum_{q=0}^{(k-j-1) \mod N_i} P_{i,p}^{(j+q) \mod N_i} \right) \right) + D_{i,p}^k + \lfloor \frac{\boldsymbol{t}}{\boldsymbol{\mathcal{P}_i}} \rfloor * \boldsymbol{\mathcal{P}_i}$
10 $\quad y_{i,t,p}^j = \sum_{k=0}^{N_i-1} y_{i,t,p}^{j,k}, \quad \forall i, j, t, p.$
11 $\quad y_{i,t,p} \geq y_{i,t,p}^j, \quad \forall i, j, t, p.$
12 $\quad \sum_{i=0}^{n-1} y_{i,t,p} \leq \mathcal{L} * \boldsymbol{t} \quad \forall t, p.$
13 and:
14 $\quad D_{i,p}^k, P_{i,p}^k, t_b, y_{i,t,p}^{j,k}, y_{i,t,p}^j, y_{i,t,p}, \mathcal{L} \in \mathbb{R}^*, x_{i,t,p}^{j,k} \in \{0, 1\}.$

In the *Deadline Assignment and Feasibility Test* MILP, the notations in bold font are constants and the other notations are variables. Lines 3 to 6 present the basic constraints introduced in Section 3. Line 7 calculates the demand of $y_{i,p,t}^{j,k}$. The interval length $\lfloor \frac{t}{\mathcal{P}_i} \rfloor$ tracks the number of cycle periods in t, and $\lfloor \frac{t}{\mathcal{P}_i} \rfloor * E_{i,p}^k$ is the demand of $\phi_{i,p}^k$ in such cycle periods. The parameter $x_{i,t,p}^{j,k}$ is restricted to be an integer value and works as a "flag" (either 0 or 1) to decide whether the demand $E_{i,p}^k$ should be added in the interval length $t - \lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i$ as shown in Figure 4. Note that all frames are released as soon as possible. The analysis of a demand in $[0, t - \lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i]$ is equal to the one in $[\lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i, t]$. The "flag" $x_{i,t,p}^{j,k}$ is decided by the constraints in Lines 8 and 9. Line 8 is

the constraint function that decides the value of $x^{j,k}_{i,t,p}$. The length t_b in Line 9 is the summation of the previous periods $\lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i$ and the distance from the starting j'th frame to k'th frame $\left(\sum_{p=0}^{Q-1} \left(\sum_{q=0}^{(k-j-1) \mod N_i} P_{i,p}^{(j+q) \mod N_i} \right) \right) + D^k_{i,p}$. The length t_b ensures the sequence of real frames in distributed systems. That is, since the frames before $\phi^k_{i,p}$ may be empty frames on the processor p, we add all the periods of the j'th frame to $k-1'$th frame in all processors. For example, the length $t_b = \left(\sum_{p=0}^{Q-1} P^1_{i,p} + \sum_{p=0}^{Q-1} P^2_{i,p} \right) + D^3_{i,p} + \lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i$ if we consider the interval starting with the arrival of the first frame and ending with the deadline of the third frame in the end-to-end flow τ_i. In the inequality of Line 8, the lengths t_b and t decide whether the demand of the k'th frame in length $t - \lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i$ will be added to $y^{j,k}_{i,t,p}$. The constant *realmin* is the smallest representable positive number. When $t \geq t_b$, the flag $x^{j,k}_{i,t,p}$ must be 1 and the demand $x^{j,k}_{i,t,p} * E^k_{i,p}$ contributes to $y^{j,k}_{i,t,p}$. When $t < t_b$, the flag $x^{j,k}_{i,t,p}$ can be either 0 or 1. However, the demand $y^{j,k}_{i,t,p}$ is overestimated when $x^{j,k}_{i,t,p} = 1$. The solver MILP tends to choose 0 for $x^{j,k}_{i,t,p}$ because of the smaller demand, and the details are shown in Lemma 1. Note that the inequality in Line 8 is always correct when $x^{j,k}_{i,t,p}$ is 1 and $t \geq t_b$, and when $x^{j,k}_{i,t}$ is 0 and $t < t_b$.

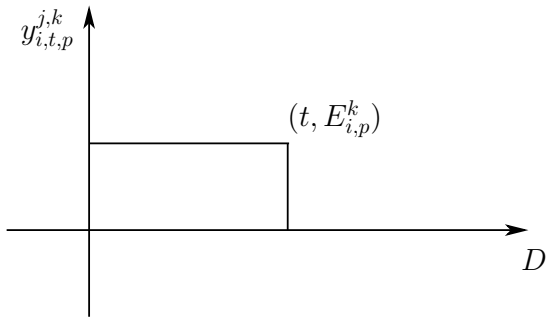


Figure 4: The demand $y^{j,k}_{i,t,p}$ in this figure is calculated when t is smaller than one cycle period by combining Lines 7, 8 and 9. When the deadline of frame $\phi^k_{i,p}$ ends inside the interval length t, the demand $y^{j,k}_{i,t,p}$ is $E^k_{i,p}$. Otherwise, the demand $y^{j,k}_{i,t,p}$ is zero.

In Line 10, the demand $y^j_{i,t,p}$ of task τ_i starts from the j'th frame. In Line 11, the demand $y_{i,t,p}$ is the maximum demand for $\tau_{i,p}$ over all possible starting frames. At last, the demand of all tasks $\sum_{i=0}^{n-1} y_{i,t,p}$ has to be less than the supply bound function for all interval lengths t and processors p as shown in Equation 1; otherwise, the system is not schedulable. In Line 12, $\mathcal{L}$ is set to indicate the degree of schedulability of the system. If the system is schedulable, then $\mathcal{L} \leq 1$.

In the setting of our MILP, the variables $D^k_{i,p}$, $P^k_{i,p}$, t_b, $y^{j,k}_{i,t,p}$, $y^j_{i,t,p}$, $y_{i,t,p}$, and $\mathcal{L}$ are free variables. The number of all variables is pseudo-polynomial bounded. The flag $x^{j,k}_{i,t,p}$ is restricted to be an integer variable that is either 0 or 1. The relationship among the variables is summarized in Figure 5. The boxes with solid lines contain free variables and the boxes with dotted lines contain constants. The arrows show the dependable relationships and the integers on the arrows indicate the number of lines in the MILP. For example, Lines 6 to 9 show that the constant $\mathcal{P}_i$ has an effect on the variables $P^k_{i,p}$, t_b, $x^{j,k}_{i,t,p}$ and $y^{j,k}_{i,t,p}$. All variables are connected and constrained in MILP. Eventually, minimizing $\mathcal{L}$ also minimizes the total demand $\sum_{i=0}^{n-1} y_{i,t,p}$.

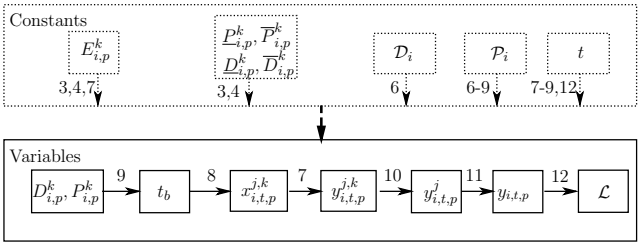


Figure 5: Relationship among the parameters.

In our dGMF-PA model for distributed systems, we prove that our MILP is a necessary schedulability test in general, and the MILP is also a sufficient and necessary schedulability test for integer parameters in Lemma 1 and Theorem 1.

LEMMA 1. *The value of $y^{j,k}_{i,t,p}$ in the MILP is the exact worst-case demand of frames $\phi^k_{i,p}$ over an interval of length t when the first frame of τ_i to arrive in the interval is j'th frame. (with respect to the deadline assigned to each frame of $\tau_{i,p}$ by the MILP).*

PROOF. If the j'th frame is not assigned on the processor p, the demand $y^{j,k}_{i,t,p}$ is the exact worst-case demand which is zero. The proof is straightforward since the frame does not execute on the processor p.

When the j'th frame is assigned on the processor p, it is trivial that $y^{j,k}_{i,\lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i,p}$ is the exact demand $\lfloor \frac{t}{\mathcal{P}_i} \rfloor * E^k_i$ in the time interval length $\lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i$. We will prove that the worst-case demand $y^{j,k}_{i,t',p} = x^{j,k}_{i,t,p} * E^k_{i,p}$ is exact in the interval length $t' = t - \lfloor \frac{t}{\mathcal{P}_i} \rfloor * \mathcal{P}_i$. Worst-case means that the interval length t starts at the release time of the j'th frame and all succeeding frames are released as soon as possible. We will show that $y^{j,k}_{i,t',p}$ is an upper bound and a lower bound on the demand. That is, the demand $y^{j,k}_{i,t,p}$ is exact. For simplicity, we refer to $y^{j,k}_{i,t',p}$ ($x^{j,k}_{i,t',p}$) as y (x).

Assume that $t'' = \left(\sum_{p=0}^{Q-1} \left(\sum_{q=0}^{(k-j-1) \mod N_i} P_{i,p}^{(j+q) \mod N_i} \right) \right) + D^k_{i,p}$, there are also two situations: when $0 \leq t' < t''$ and $t'' \leq t' < \mathcal{P}_i$. Note that t' is smaller than $\mathcal{P}_i$ from definition. When $0 \leq t' < t''$, x can be zero or one from the MILP. Since we minimize $\mathcal{L}$ in MILP, y is also minimized to take the value zero (by $x = 0$). When $t'' \leq t' < \mathcal{P}_i$, y has to be $E^k_{i,p}$ to satisfy the constraints in Lines 7 to 9 of our MILP.

When $0 \leq t' < t''$, $y = 0$. The demand y is a lower bound since no demand takes negative values. We prove that y

is an upper bound by contradiction. If there exist $y' > y$, $y' = E^k_{i,p}$ since x can only take an integer value one or zero. In this case, $t' \geq t''$ and get a contradiction with $0 \leq t' < t''$. y is an upper bound and a lower bound when $0 \leq t' < t''$.

When $t'' \leq t' < \mathcal{P}_i$, $y = E^k_{i,p}$ since $x = 1$. The proof of the lower bound is similar to the proof of the upper bound when $t'' \leq t' < \mathcal{P}_i$. That is the demand cannot be smaller than y; otherwise, t' will be smaller than t''. We prove that the demand y is an upper bound by contradiction. Assume that the demand y' is the upper bound which is larger than the demand $y = E^k_{i,p}$. If $y' > E^k_{i,p}$, the corresponding interval length t' has to be larger than $\mathcal{P}_i$. This is a contradiction since $t'' \leq t' < \mathcal{P}_i$. y is an upper bound and a lower bound when $t'' \leq t' < \mathcal{P}_i$.

In total, the demand $y^{j,k}_{i,t,p}$ is the exact worst-case demand for the frame $\phi^k_{i,p}$ over an interval of length t when the first frame of $\tau_{i,p}$ to arrive in the interval is the j'th frame. □

THEOREM 1. *For arbitrary, real-valued parameters, our MILP is a necessary feasibility test. When the period and deadline parameters are integers (i.e., $D^k_{i,p}$, $P^k_{i,p} \in \mathbb{N}$, $\forall$ i, k and p), the MILP is an exact feasibility test.*

PROOF. It is straightforward to prove that our MILP is a necessary feasibility test in general. If a distributed system is feasible, the worst-case demand ($\sum_{i=0}^{n-1} y_{i,t,p}$) of all tasks over any interval length t must be smaller than t in any processor p.

In Lemma 1, we have proved that $y^{j,k}_{i,t,p}$ in the MILP is the exact worst-case demand of frames $\phi^k_{i,p}$ over an interval of length t when the first frame of τ_i to arrive in the interval is the j'th frame. $y^j_{i,t,p}$ is thus the exact demand of task $\tau_{i,p}$ over length t starting from the j'th frame, and $y_{i,t,p}$ is the exact worst-case demand of $\tau_{i,p}$ over length t. $\sum_{i=0}^{n-1} y_{i,t,p} \leq \boldsymbol{t}$ is a sufficient feasibility test when $D^k_{i,p}$, $P^k_{i,p}$, t $\in \mathbb{N}$. The algorithm is exact when the frame deadline and period parameters are integers, since it can be easily shown that the dbf changes value in this case only at integer times; thus, the MILP exactly checks all the relevant time intervals. Note that our MILP in general is not a sufficient feasibility test when this integer constraint is removed since it does not check all real values in the range $[0, H]$. □

Due to the fact that our MILP is not an exact feasibility test in general, we introduce a sufficient feasibility test in general in Section 5. The sufficient feasibility test is an approximation algorithm based on our MILP, where the running time is greatly reduced.

5. THE APPROXIMATION ALGORITHM BASED ON OUR MILP

In the previous section, we have built our MILP which can select the relative deadlines of dGMF-PA tasks under EDF scheduling. The method also indicates a necessary feasibility test at the same time. However, solving an MILP is NP-hard in general. Furthermore, the feasibility of selecting deadlines in the dGMF-PA model is coNP-hard as the problem can be can be trivially transformed from the feasibility test of sporadic tasks [14]. In this section, we will modify the MILP to obtain an approximation algorithm based on

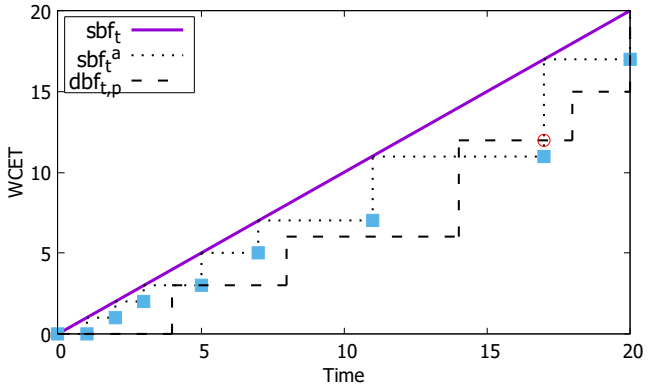


Figure 6: The staircase function drawn in dashed line is an example of demand $\mathsf{dbf}_{t,p} = \sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p}$ on processor p. The x-axis values of square points on sbf^a_t are in the set T_a, which are enough to generate a sufficient schedulablility test. In this example, the total demand $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p} \leq \mathsf{sbf}_t$ at all time interval length t. But, the demand $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p} > \mathsf{sbf}^a_t$ is shown at the red circle.

reducing the number of time interval lengths being tested[3]. We also show that the speed-up factor of our approximation algorithm is $1 + \epsilon$ with respect to the exact schedulability test of dGMF-PA tasks under EDF scheduling. We have introduced an approximation algorithm under the GMF-PA model [25] and such similar technique can be traced back to admission control for the arbitrary demand curves [11].

Assume that the number of time interval lengths being tested in MILP is H (defined in Section 4), and the one in the approximation algorithm is H_a. The set of time interval lengths in MILP and the approximation are $T = \{1, 2, 3, ..., H\}$ and T_a, respectively. The supply bound function used in MILP is shown in Equation 2.

$$\mathsf{sbf}_t = t. \tag{2}$$

Since the number of variables and equations in MILP depends on H, the size of MILP grows quickly when H grows. We propose an approximation method based on reducing the number of time intervals. We start from the initial time interval length t_0. The increasing rate is $\epsilon > 0$. We choose the interval length by the increasing rate; thus, $T_a = \{t_0, t_0 * (1+\epsilon), t_0 * (1+\epsilon)^2, ..., t_0 * (1+\epsilon)^{H_a - 2}, H\}$. Note that the $H_a - 2'$th element is no larger than H, and we add H at the end as the $H_a - 1'$th element. Also, the increasing rate between the last two elements is no larger than ϵ. For example, the set T_a is $[1, 1.5, 2.25, 3.375, ..., 17.0859375, 20]$ for $H = 20, t_0 = 1$ and $\epsilon = 0.5$. The supply sbf^a_t in the approximation algorithm is shown in Equation 3.

$$\mathsf{sbf}^a_t = \begin{cases} 0, & 0 \leq t \leq t_0 \\ t_0 * (1+\epsilon)^k, & t_0 * (1+\epsilon)^k < t \leq t_0 * (1+\epsilon)^{k+1} \\ H, & t = H \end{cases} \tag{3}$$

[3]The approximation is still an MILP (and thus still potentially intractable), but a reduction in constraints leads to a significant improvement in efficiency as shown in the evaluation section.

In our sbf_t^a, the initial interval length is $t_0 = \min_{\tau_i \in \mathcal{T}} E_i^{min}$ and the range of integer k is $[1, H_{a-2}]$ in our approximation algorithm. Figure 6 shows an example of the relationship among $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p}$, sbf_t and sbf_t^a on processor p. It is straightforward to show that the number of elements in T_a is $O(\log_{1+\epsilon} H)$.

Next, we modify the general schedulability condition of Equation 1 with respect to the reduced set T_a.

THEOREM 2. *Consider any distributed task system composed of tasks $\mathcal{T}$ (e.g., dGMF-PA tasks) where the $\mathsf{dbf}_{i,t,p}$ is computable (e.g., see Peng and Fisher [25]) for any $\tau_i \in \mathcal{T}$ and $p \in \mathcal{Q}$ ($\mathcal{Q}$ is the index set of processors). Then, by checking the following modified condition:*

$$\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p} \leq \mathsf{sbf}_t^a, \quad \forall t \in T_a,\ p \in \mathcal{Q}, \tag{4}$$

where t_0 of sbf_t^a must not be larger than $\min_{i,j,p}\{\sum_{p=0}^{Q-1} D_{i,p}^j\}$. We have the following guarantee:

1. *If $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p} \leq \mathsf{sbf}_t^a$, $\forall t \in T_a$, $p \in \mathcal{Q}$, the distributed system is EDF-schedulable on unit-speed processors.*

2. *If $\exists t \in T_a$ and $p \in \mathcal{Q}$, $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p} > \mathsf{sbf}_t^a$, the system is EDF-infeasible where each processor is $\frac{1}{1+\epsilon}$-speed.*

PROOF. We first prove the sufficiency. If $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t',p} \leq \mathsf{sbf}_{t'}^a$ at an interval length $t' = t_0 * (1+\epsilon)^k$ for any $k \in H_a$ in Equation 3, all demands over the range of intervals $(\frac{t'}{1+\epsilon}, t']$ are also smaller than $\mathsf{sbf}_{t'}^a$ since the demand bound function is a monotonically increasing function. In other words, the system is schedulable on any interval length in $(\frac{t'}{1+\epsilon}, t']$ if the system is schedulable on interval length t'. The test intervals are thus reduced to the set T_a. If $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p} \leq \mathsf{sbf}_t^a$ for all $t \in T_a$ and $p \in \mathcal{Q}$, all unit-speed processors are EDF-schedulable. The distributed system composed by the processors is also EDF-schedulable, which indicates the sufficiency in the distributed system.

We prove the infeasibility on a slower processor when Equation 4 is not satisfied (equal to the proof of the "speed-up factor"). Assume $\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,p,t^*} > \mathsf{sbf}_{t^*}^a$ at time interval length t^* and processor p. It must be that $t^* > t_0$ since for all values of $t \leq t_0$, $\mathsf{dbf}_{i,t,p}$ is zero by supposition that t_0 exceeds the minimum frame relative deadline. Furthermore, it is easy to observe that for all $t \geq t_0$, the sbf_t is at most $(1+\epsilon)$ times larger than sbf_t^a. From this, we have:

$$\begin{aligned}
\max_{t>0} \frac{\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p}}{\mathsf{sbf}_t} &\geq \frac{\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t^*,p}}{\mathsf{sbf}_{t^*}} \\
&\geq \frac{\frac{\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t^*,p}}{(1+\epsilon)}}{\frac{\mathsf{sbf}_{t^*}}{(1+\epsilon)}} \\
&\geq \frac{\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t^*,p}}{\mathsf{sbf}_{t^*}^a * (1+\epsilon)} \quad \text{(By Equation 3)} \\
&\geq \frac{1}{1+\epsilon} \quad \text{(By assumption).}
\end{aligned}$$

Summing both sides of the above derived inequality, we get:

$$\begin{aligned}
\sum_{p \in \mathcal{Q}} \left(\max_{t>0} \frac{\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p}}{\mathsf{sbf}_t} \right) &\geq \frac{Q}{1+\epsilon} \\
Q * \max_{t>0, p \in \mathcal{Q}} \frac{\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p}}{\mathsf{sbf}_t} &\geq \frac{Q}{1+\epsilon} \\
\max_{t>0, p \in \mathcal{Q}} \frac{\sum_{\tau_i \in \mathcal{T}} \mathsf{dbf}_{i,t,p}}{\mathsf{sbf}_t} &\geq \frac{1}{1+\epsilon}
\end{aligned}$$

Thus, we have proved that the speed-up factor is $1+\epsilon$ over all processors in distributed systems, with respect to the exact schedulability test of dGMF-PA tasks under EDF scheduling. □

We can now apply Theorem 2 to modify the MILP to create a sufficient approximate feasibility test for the dGMF-PA task model with arbitrary, real-valued parameters. To do so, we simply limit the range of t to now be T_a for all constraints that depend upon t, and modify Line 12 of MILP to be $\sum_{i=0}^{n-1} y_{i,t,p} \leq \mathcal{L} * \frac{\boldsymbol{t}}{1+\epsilon}$. Clearly, this reduces the number of constraints by a logarithmic factor (dependent upon our choice of ϵ). We refer to this approximate assignment algorithm as MILP-ϵ.

In all, the approximate MILP is a sufficient feasibility test. The number of the time interval lengths is reduced from $O(H)$ to $O(\log_{1+\epsilon} H)$. Since the number of variables and number of equations depend on the number of time interval lengths, the running time is greatly reduced.

6. EVALUATION

We have implemented our MILP and approximation algorithm MILP-ϵ ($\epsilon > 0$) using the commercial solver GUROBI [1] in MATLAB. GUROBI is a state-of-the-art mathematical programming solver that has great performance in solving linear and mixed-integer programming problems. We compare our work with the combination (represented by HOSPA-Offset) of the deadline assignment HOSPA [26,27] and offset-based analysis under EDF scheduling [23] in the MAST suite [2].

In order to generate a fair comparison with HOSPA-Offset, we set $D_{i,p}^k = P_{i,p}^k$ (Line 5 in MILP will be automatically satisfied) and $\mathcal{D}_i = \mathcal{P}_i$. The variables $D_{i,p}^k, P_{i,p}^k, \mathcal{D}_i$, and $\mathcal{P}_i$ are reduced to $D_{i,p}^k$ and $\mathcal{P}_i$ for all i, k and p. In this case, the end-to-end deadline of $\tau_{i,p}$ is $\mathcal{P}_i$. Because HOSPA-Offset has no frame constraints, we set $\underline{D}_{i,p}^k = E_{i,p}^k$ and $\overline{D}_{i,p}^k = \mathcal{P}_i$ for all frames. The constraints from Lines 3 to 6 of our MILP and MILP-ϵ thus become:

1. $\boldsymbol{E_{i,p}^k} \leq D_{i,p}^k, \quad \forall i,k,p.$

2. $\sum_{p=0}^{Q-1} \sum_{k=0}^{N_i-1} D_{i,p}^k \leq \boldsymbol{\mathcal{P}_i}, \quad \forall i.$

We follow the similar setting of the previous paper [26] to randomly generate end-to-end flows (tasks). There are five processors and eight tasks in the distributed system. Each task contains five frames which are randomly assigned on the processors in the distributed system. There are nine system utilization levels (100%, 125%, 150%,..., 300%) each of which contains fifty distributed systems. In each system,

we use the UUniFast algorithm [8] to generate the utilization of each task and the execution time of each frame. We test the schedulability ratio (the number of schedulable systems over the number of total systems) and average running time of the task sets on each utilization level.

For the reason that our MILP is not scalable in general, we first generate Figure 7(a) and 7(b) in which tasks have small cycle periods that are randomly chosen in [1,10]. Note that each curve of MILP is a characterization of an "upper bound" on the best we can hope for in our model. MILP-0.1 and MILP-0.3 have higher schedulability ratio than HOSPA-Offset as shown in Figure 7(a), but have longer running time as shown in Figure 7(b). MILP-0.1 can schedule at most 44 % more than HOSPA-Offset when $U_{cap} = 2$, and MILP-0.3 can schedule at most 18 % more than HOSPA-Offset when $U_{cap} = 2$. MILP is at most 19.1 times slower than HOSPA-Offset, MILP-0.1 is at most 4.8 times slower than HOSPA-Offset, and MILP-0.3 is at most 0.5 times slower than HOSPA-Offset.

In order to generate a set of tasks with larger cycle periods, we have the experiments shown in Figure 7(c) and 7(d). The cycle period of each task is randomly chosen in [1,1000]. Figure 7(c) shows that any experiment with $\epsilon \leq 0.3$ will generate better schedulability ratio than HOSPA-Offset. MILP-0.3 can schedule at most 18 % more than HOSPA-Offset when $U_{cap} = 2$, and MILP-0.3 uses at most around 303 seconds more than HOSPA-Offset when $U_{cap} = 1.75$.

In all, our MILP and MILP-ϵ algorithms always yield higher schedulability ratio. The running time of the combined technique HOSPA-Offset is shorter in general; however, our MILP-ϵ is not worse by much and still efficient enough.

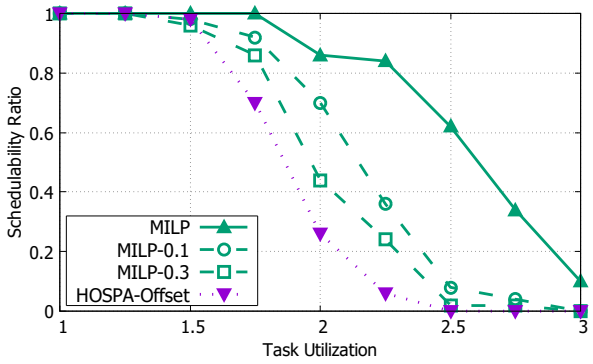
7. CONCLUSION

Upon the flexible GMF-PA model in uniprocessor systems, we propose the dGMF-PA model in distributed real-time systems. The relative deadlines of frames in end-to-end flows can be flexibly chosen in our dGMF-PA model, using the mixed-integer linear programming (MILP). Our MILP-based algorithm is an exact feasibility test when parameters are integers, and a necessary feasibility test in general. In order to reduce the running time of the MILP algorithm and give an sufficient schedulability test (in general), we propose an approximation algorithm MILP-ϵ based on the supply bound function. The number of time interval lengths is bounded by a logarithmic function of the task system parameters. We prove that the MILP-ϵ is a sufficient feasibility test. Exhaustive experiments have shown that our algorithms have improved the schedulability ratio compared to the previous results.

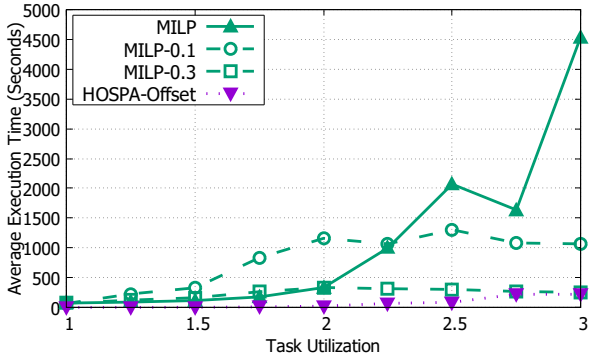
In the future, we will work to further improve the efficiency of our algorithm by considering other optimization techniques that remove the integer requirement of our MILP. We will also apply our algorithms to more complex task sets (e.g., DAG tasks). Our overall goal is an algorithm that can be used as an online optimization technique for determining parameters in an interactive real-time distributed system design framework.

Acknowledgments

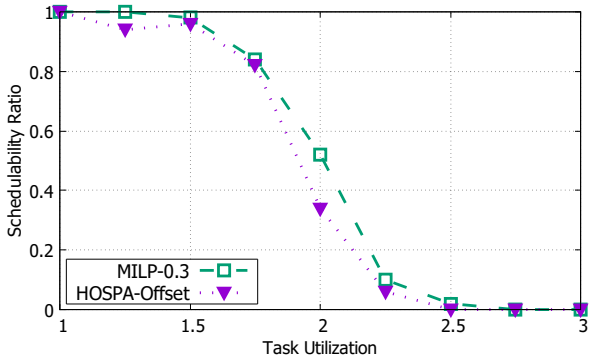
We are grateful to the anonymous reviewers whose comments have to significantly improve our paper. This research



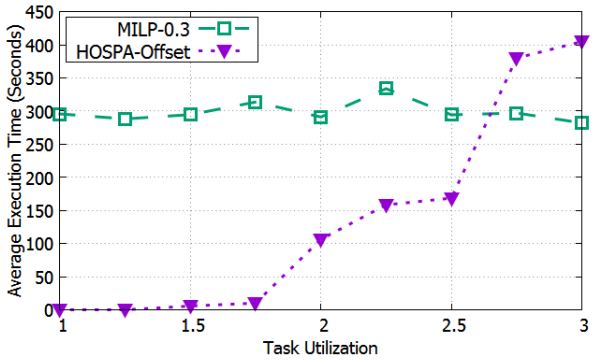
(a) $\mathcal{P}_i \in [1, 10]$



(b) $\mathcal{P}_i \in [1, 10]$



(c) $\mathcal{P}_i \in [1, 1000]$



(d) $\mathcal{P}_i \in [1, 1000]$

Figure 7: The figures show the schedulability ratio and average running time over task utilization from one to three. $\mathcal{P}_i \in [1, 10]$ is in Figures 7(a) and 7(b), and $\mathcal{P}_i \in [1, 1000]$ is in Figures 7(c) and 7(d).

has been supported in part by the US National Science Foundation (CNS Grant Nos. 0953585, 1205338, 1618979 & 1618185).

8. REFERENCES

[1] Gurobi: The state-of-the-art mathematical programming solver. http://www.gurobi.com/.

[2] Mast: Modeling and analysis suite for real-time applications. http://mast.unican.es/.

[3] B. Andersson. Schedulability analysis of generalized multiframe traffic on multihop-networks comprising software-implemented ethernet-switches. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, April 2008.

[4] S. Baruah. The non-cyclic recurring real-time task model. In *Proceedings of the 31st IEEE Real-Time Systems Symposium*, pages 173–182, Nov 2010.

[5] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *Real-Time Systems*, pages 5–22, 1999.

[6] S. K. Baruah. Dynamic- and static-priority scheduling of recurring real-time tasks. *Real-Time Syst.*, 24(1):93–128, Jan. 2003.

[7] S. K. Baruah, R. R. Howell, and L. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.

[8] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, pages 129–154, 2005.

[9] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Compututers*, pages 289–302, 2002.

[10] T. Chantem, X. Wang, M. Lemmon, and X. Hu. Period and deadline selection for schedulability in real-time systems. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS)*, pages 168–177, July 2008.

[11] F. Dewan and N. Fisher. Efficient admission control for enforcing arbitrary real-time demand-curve interfaces. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium*, pages 127–136, Washington, DC, USA, 2012. IEEE Computer Society.

[12] S. Ding, H. Tomiyama, and H. Takada. Scheduling algorithms for i/o blockings with a multi-frame task model. In *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug 2007.

[13] P. Ekberg, N. Guan, M. Stigge, and W. Yi. An optimal resource sharing protocol for generalized multiframe tasks. *Journal of Logical and Algebraic Methods in Programming*, 84(1):92 – 105, 2015.

[14] P. Ekberg and W. Yi. Uniprocessor feasibility of sporadic tasks remains coNP-complete under bounded utilization. In *Proceedings of the 36th IEEE Real-Time Systems Symposium (RTSS)*, 2015.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., NY, USA, 1990.

[16] P. Jayachandran and T. Abdelzaher. Delay composition in preemptive and non-preemptive real-time pipelines. *Real-Time Syst.*, 40(3):290–320, Dec. 2008.

[17] J. Liu. Real-Time Systems. *Prentice Hall*, 2000.

[18] J. Mäki-Turja and M. Nolin. Efficient implementation of tight response-times for tasks with offsets. *Real-Time Systems*, 40(1):77–116, 2008.

[19] S. Matic and T. A. Henzinger. Trading end-to-end latency for composability. In *Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS)*, pages 12 pp.–110, Dec 2005.

[20] A. Mok and D. Chen. A multiframe model for real-time tasks. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, pages 22–29, Dec 1996.

[21] N. Moyo, E. Nicollet, F. Lafaye, and C. Moy. On schedulability analysis of non-cyclic generalized multiframe tasks. In *Proceedings of the 22nd Euromicro Conference Real-Time Systems (ECRTS)*, pages 271–278, July 2010.

[22] J. C. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings of the 19th IEEE Real-Time Systems Symposium*, pages 26–37, Dec 1998.

[23] J. C. Palencia and M. G. Harbour. Offset-based response time analysis of distributed systems scheduled under edf. In *Proceedings of 15th Euromicro Conference on Real-Time Systems*, pages 3–12, July 2003.

[24] R. Pellizzoni and G. Lipari. Improved schedulability analysis of real-time transactions with earliest deadline scheduling. In *Proceedings of the 11th IEEE Real Time and Embedded Technology and Applications Symposium*, pages 66–75, March 2005.

[25] B. Peng and N. Fisher. Parameter adaption for generalized multiframe tasks and applications to self-suspending tasks. In *Proceedings of the 22nd Embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2016.

[26] J. M. Rivas, J. J. Gutiérrez, J. C. Palencia, and M. G. Harbour. Schedulability analysis and optimization of heterogeneous edf and fp distributed real-time systems. In *Proceedings of the 23rd Euromicro Conference on Real-Time Systems (ECRTS)*, pages 195–204, July 2011.

[27] J. M. Rivas, J. J. Gutiérrez, J. C. Palencia, and M. G. Harbour. Deadline assignment in edf schedulers for real-time distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, Oct 2015.

[28] N. Tchidjo Moyo, E. Nicollet, F. Lafaye, and C. Moy. Real time scheduling analysis for DSP base band processing in multi-channel SDR set. In *Proceedings of the SDR Forum Technical Conference*, Washington, United States, Dec. 2009.

[29] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming - Parallel processing in embedded real-time systems.*, 40(2-3):117–134, Apr. 1994.