Dimensions of fixed-priority aperiodic servers

Abhishek Singh
abhishek.s@wustl.edu
Washington University in St. Louis
St. Louis, Missouri, U.S.A.

Sanjoy Baruah
baruah@wustl.edu
Washington University in St. Louis
St. Louis, Missouri, U.S.A.

ABSTRACT

We identify the budget and the utilization of an aperiodic server as vital attributes that affect its performance. Based on this observation, we formulate an optimization problem in which we are given a minimum budget for multiple servers running at the same priority, and the objective is to find the dimensions (budgets and periods) of these servers to maximize their cumulative utilization. We propose a linear-time algorithm for solving the problem if priorities are rate-monotonic, periods are harmonic, and deadlines are equal to periods. We also propose mixed-integer nonlinear programs for the general problem when these simplifying assumptions are lifted. Finally, we discuss issues arising when implementing multiple servers at the same priority, and we show how to modify the specifications of servers to address these issues.

CCS CONCEPTS

• **Computer systems organization** → **Real-time operating systems**; **Embedded software**.

KEYWORDS

aperiodic servers, fixed priority, harmonic, algorithm design, optimization, MINLP

ACM Reference Format:
Abhishek Singh and Sanjoy Baruah. 2023. Dimensions of fixed-priority aperiodic servers. In *The 31st International Conference on Real-Time Networks and Systems (RTNS 2023), June 07–08, 2023, Dortmund, Germany*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3575757.3593639

1 INTRODUCTION

Safety-critical systems can often be modeled as a collection of preemptible priority-driven[1] *hard real-time tasks* (HRT tasks) and *aperiodic tasks*, where the HRT tasks are recurrent and have strict timing requirements, and the aperiodic tasks are isolated and have loose timing requirements. For such hybrid collections, an *aperiodic server* is often used to serve aperiodic tasks at a high priority while simulating the timing behavior of one or more artificial HRT tasks at that priority. The use of aperiodic servers results in smaller response times for the aperiodic tasks; moreover, traditional real-time scheduling theory can be used to analyze the system since it effectively contains only HRT tasks (after aperiodic tasks are replaced by artificial HRT tasks). The parameters of the artificial HRT tasks, which are called the *dimensions* of the aperiodic server, are chosen so that the original HRT tasks meet their timing requirements and the response times of the aperiodic tasks are reduced; the problem of choosing dimensions to meet the above objectives is called *dimensioning* the server.

We want to dimension servers when the original HRT tasks have *fixed priorities* (tasks are assigned priorities that remain constant for all behaviors) and *constrained deadlines* (for each task, its deadline is at most its period; see Section 1.1 for terminology), are preemptible, and run on a uniprocessor; we refer to fixed-priority constrained-deadline preemptive uniprocessor systems as simply *FP systems*. The theory for the timing analysis of FP systems is rooted in ideas like the critical instant and response time analysis (RTA) [2, 3, 14, 17]; Section 2 explains the elements of the theory that we utilize in this work. This theory underpins a rich framework for the design, implementation, and analysis of safety-critical real-time systems that has won widespread acceptance in industrial practice [11, 12, 22]; for instance, support for this framework is included in the IEEE POSIX standard application program interface (API) for operating system services [1].

We refer to servers for FP systems as *FP servers*. Notable FP servers include the polling server, the sporadic server, the deferrable server, and the priority exchange server [15, 23, 26]. The deferrable server and the sporadic server are arguably the more well-studied servers [7, 8, 24] because both servers are bandwidth-preserving and have performed similarly in empirical evaluations [4]. Although the initial motivation for FP servers was enhancing aperiodic responsiveness in systems containing HRT and aperiodic tasks, FP servers have also been used in other contexts such as resource reservations for multimedia applications [19, 20] and hierarchical scheduling in open environments [8, 9, 13, 16, 18, 21]. Our aim is to dimension FP servers in the original context, i.e., to enhance aperiodic responsiveness. Before describing the existing work on dimensioning servers, its limitations, and our proposed approach, we must introduce some properties of FP servers.

1.1 FP systems and servers: model and terms

Our system contains *sporadic tasks*, which are a type of HRT tasks, and aperiodic tasks. The HRT subsystem, which is an FP system, is represented as a list

$$\Gamma = \langle \tau_1, \tau_2, \ldots, \tau_n \rangle,$$

where τ_i is a sporadic task, and the tasks are listed in decreasing order of priority.[2] τ_i receives at most one request in any interval of

[1] In preemptive priority-driven systems, at any instant, a pending task with the highest priority amongst all pending tasks is picked to run on the processor.

[2] We assume that $\langle e_1, \ldots, e_n \rangle$ is a list of n elements; $\langle\rangle$ is an empty list; and $l_1 \circ l_2$ is the concatenation of lists l_1 and l_2.

RTNS 2023, June 07–08, 2023, Dortmund, Germany

ACM ISBN 978-1-4503-9983-8/23/06.
https://doi.org/10.1145/3575757.3593639

length T_i (T_i is the *period* of τ_i); τ_i must complete the response to the request within D_i units of time (D_i is the *relative deadline* of τ_i); the execution of any single response for τ_i takes at most C_i units of time on the processor (C_i is called the *wcet* (worst-case execution time) of τ_i). Thus, τ_i may be represented as the triple (C_i, T_i, D_i), or the pair (C_i, T_i) if the deadline is equal to the period. If τ_i does not miss its deadlines in any behavior of the system, then we say that τ_i is *schedulable* in the system. If τ_i is schedulable for all $i \in [n]$[3], then we say that the system is schedulable.

For any $i \in [n]$, we use Γ_i (resp., $\Gamma_n - \Gamma_i$) to refer to the high (resp., low) subsystem containing tasks $\langle \tau_1, \tau_2, \ldots, \tau_i \rangle$ (resp., $\langle \tau_{i+1}, \tau_{i+2}, \ldots, \tau_n \rangle$). The FP server serves the aperiodic tasks at a high priority k to reduce their response times. We assume that $k \in [n+1]$, all tasks in the FP subsystem Γ_{k-1} have a higher priority than the server, and all tasks in the FP subsystem $\Gamma_n - \Gamma_{k-1}$ have lower priority than the server. Recall that one of the objectives of the server is to ensure that the system is schedulable; we refer to a server that meets this objective as a *feasible* server. The schedulability of the FP subsystem Γ_{k-1} is unaffected by the server because it has higher priority than the server and the server cannot interfere with its timing behavior. The schedulability of the FP subsystem $\Gamma_n - \Gamma_{k-1}$, on the other hand, is affected by the server. Thus, the feasibility of a server is equivalent to the schedulability of the lower FP subsystem $\Gamma_n - \Gamma_{k-1}$.

The server has a *(execution-time) budget*[4] B and a *period* P. The server (k, B, P) simulates the timing behavior of m sporadic tasks, denoted $\kappa_1, \ldots, \kappa_m$. The m tasks have priority k, cumulative wcet B, period P, and (implicit) deadline P. For instance, the sporadic server (k, B, P) simulates the timing behavior of B sporadic tasks with wcet 1 and period P, and the polling server (k, B, P) simulates the timing behavior of 1 sporadic task with wcet B and period P. Internally, the server does a lot of bookkeeping to ensure that the simulation is valid; for instance, the server keeps track of how much budget has been consumed by aperiodic tasks, it ensures that the aperiodic tasks do not consume more budget than what is available, and it decides when the budget should be replenished next. The consumption and replenishment rules vary between servers depending on the collection of tasks they aim to simulate (we discuss the rules for sporadic servers in Section 6).

Externally, from the perspective of the FP scheduler, the server appears to be a collection of sporadic tasks with the same priority and is indistinguishable from the true sporadic tasks in Γ. Thus, we must choose the parameters of $\kappa_1, \ldots, \kappa_m$ so that the following system is schedulable:

$$\Gamma_{k-1} \circ \langle \kappa_1, \ldots, \kappa_m \rangle \circ (\Gamma_n - \Gamma_{k-1})$$

Application of elementary results from FP scheduling theory allows the system to be simplified to

$$\Gamma_{k-1} \circ \langle (B, P) \rangle \circ (\Gamma_n - \Gamma_{k-1}).$$

As mentioned before, Γ_{k-1} is not affected by the server, and we can focus on the subsystem $\Gamma_n - \Gamma_{k-1}$. The next theorem follows from these observations.

[3]We assume that $[n]$ denotes the set $\{1, 2, \ldots, n\}$ and $[0] = \emptyset$.
[4]The term *capacity* is also used to refer to the budget by some authors.

Theorem 1.1. *Let Γ be an FP system. The server (k, B, P) is feasible if and only if $\Gamma_n - \Gamma_{k-1}$ is schedulable in the larger system*

$$\Gamma_{k-1} \circ \langle (B, P) \rangle \circ (\Gamma_n - \Gamma_{k-1}).$$

It can be argued that the server task should also be schedulable to provide quality-of-service guarantees to the aperiodic tasks; we choose to work with a definition of feasibility that is only tied to the schedulability of the true HRT tasks. The description of FP servers in the above paragraph is a valid abstraction of the polling server, the sporadic server, and the priority-exchange server but it is not true for the deferrable server, which simulates a self-suspending task. Thus, our results are applicable to polling server, the sporadic server, the priority-exchange server, and any other server which matches the above description.

1.2 Previous work on dimensioning FP servers

Dimensioning the server involves selecting feasible dimensions k, B, P that reduce response times of aperiodic tasks as much as possible. In the early stages of research on FP servers, dimensions were chosen rather simply: rate-monotonic priority assignment[5] was assumed, k was chosen to be equal to the highest priority, i.e., $k = 1$, P was chosen to be equal to the smallest period T_1, and B was chosen to equal its maximum feasible value since k and P were fixed [15, 23]. A few years later, researchers observed that sometimes choosing a smaller period than T_1 can result in a server with larger utilization [26]; the ratio B/P, which is called the *utilization* or *bandwidth* of the server, is a measure of the rate at which the server processes aperiodic requests. More details about the limitations in these works may be found in the research of Bernat and Burns [4, Secs. 3.2, 5]. Bernat and Burns investigated the efficacy of sophisticated dimensioning schemes for sporadic and deferrable servers by carrying out simulations on a large number of synthetic task systems, and concluded that aperiodic responsiveness is enhanced by ensuring that the server has a high priority, a large budget and a large utilization [4, Sec. 4.4].

The intuition for the above recommendations can be explained by considering two types of scenarios. An FP server shines in scenarios where an aperiodic task arrives and is served immediately because the server has a high priority and a large enough budget to accommodate the task's execution time; a larger budget means that it can serve aperiodic tasks with larger execution times immediately without waiting for replenishment. In scenarios where aperiodic tasks keep arriving and the server is continuously busy, a larger utilization restricts the backlog of pending aperiodic work to smaller values. Bernat and Burns propose a heuristic for dimensioning servers so that a large budget and a large utilization are achieved [4, Sec. 4.4]:

> The best performance can be generally achieved by selecting the capacity [budget] that corresponds to the local maxima of $U_s(c)$ [utilization as a function of budget] closer to the maximum possible capacity [budget].

It is evident that the above heuristic is biased towards achieving the largest budget or at least getting close to it (we will address this

[5]Rate-monotonic priority assignment is a type of fixed priority assignment in which tasks with smaller periods have higher priorities.

limitation, amongst other limitations, in the next subsection). We do not know of any other work where FP servers are dimensioned with the objectives of maximizing the budget and the utilization.

1.3 Some observations and the problem statement

A larger utilization can often be assigned to a server by using a smaller period, while a larger budget can often be assigned to a server by using a larger period. For instance, if we try to insert a server at the highest priority ($k = 1$) into the system $\langle(1, 5), (3, 10)\rangle$, then we discover that

(i) The maximum budget is 4, and the maximum utilization for this budget is $4/9 \approx 0.44$, using the server $(4, 9)$.
(ii) The maximum utilization is 0.5, and the maximum budget for this utilization is 2.5, using the server $(2.5, 5)$.

Thus, the two objectives of maximizing the budget and maximizing the bandwidth of the server are at odds with each other. Details about how to compute the above servers are provided in Section 3.

In the above example, we can add two servers $(1, 5)$ and $(3, 10)$ at the highest priority so that they serve aperiodic requests in concert with each other. After the addition, the full system is given by $\langle(1, 5), (3, 10), (1, 5), (3, 10)\rangle$; the upper half of the system contains the artificial tasks that are simulated by the server, and the lower half of the system contains the original sporadic tasks. It may be verified that the full system is schedulable and hence the parameters for the two servers are feasible. Collectively, the two servers have budget 4 and utilization 0.5, and hence they are a better choice than both servers $(4, 9)$ and $(2.5, 5)$. Thus, to improve our chances of getting larger budget and utilization values, we should consider using multiple servers at the same priority.

In this research, we ask the following question:

> Given an FP system Γ, what is the maximum cumulative utilization for a finite collection of servers that is feasible at priority k and has a minimum cumulative budget $B_{\min}$?

Let the collection of servers simulate the following collection of sporadic tasks:

$$\langle(b_1, p_1), (b_2, p_2), \ldots, (b_m, p_m)\rangle$$

All m servers run at the same priority k but a concrete implementation of such a collection of servers may decide to give preference to one server over another when both servers have available budget and an aperiodic task is pending. In our problem, the objective is to find $b_1, p_1, \ldots, b_m, p_m$, where m is arbitrary, that maximize

$$\sum_{j\in[m]} b_j/p_j$$

subject to the following constraints:

(i) $\sum_{j\in[m]} b_j \geq B_{\min}$,
(ii) $\sum_{j\in[m]} b_j \leq p_1 \leq p_2 \leq \cdots \leq p_m$, and
(iii) the collection of servers $(k, b_1, p_1), (k, b_2, p_2), \ldots, (k, b_m, p_m)$ are feasible.

The first constraint says that the cumulative budget is at least $B_{\min}$; the second constraint imposes a linear order on the periods without loss of generality, and its says that the smallest period p_1 is at least $B_{\min}$. [6] Using Theorem 1.1, the third constraint is equivalent to the schedulability of the subsystem $\Gamma_n - \Gamma_{k-1}$ in the larger system

$$\Gamma_{k-1} \circ \langle(b_1, p_1), (b_2, p_2), \ldots, (b_m, p_m)\rangle \circ (\Gamma_n - \Gamma_{k-1}).$$

We call this problem Dimensioning. An instance of Dimensioning is given by the triple $(\Gamma_n, B_{\min}, k)$.

If a system designer wants to maximize the utilization for a given set of priorities K and a minimum budget $B_{\min}$, then they can explore the space of feasible server parameters by solving the Dimensioning instances in $\{(\Gamma, B_{\min}, k) \mid k \in K\}$. The optimal utilization values for the instances will help the designer to select the server parameters $k, b_1, p_1, \ldots, b_m, p_m$ that best suit their needs.

2 MORE BACKGROUND & ASSUMPTIONS

We assume that T_i, D_i, and C_i are rational values. The ratio $U_i = C_i/T_i$ is called the *utilization* of τ_i. rbf_i, the *request-bound function* of subsystem Γ_i, is defined as follows:

$$\mathrm{rbf}_i(t) = \sum_{j\in[i]} \left\lceil \frac{t}{T_j} \right\rceil C_j \tag{1}$$

$\mathrm{rbf}_i(t)$ is the maximum cumulative execution requirement of the tasks in Γ_i corresponding to requests that arrive in any interval of length t[7].

Theorem 2.1 ([10, 14]). *If Γ is a preemptive constrained-deadline fixed-priority uniprocessor system, then Γ is schedulable if and only if for all $i \in [n]$*

$$\exists t \in (0, D_i] : \mathrm{rbf}_i(t) \leq t. \tag{2}$$

Response time analysis (RTA) [10] solves the problem

$$\min\{t \in (0, D_i] \mid \mathrm{rbf}_i(t) \leq t\} \tag{3}$$

by using a fixed-point iteration approach. RTA starts with an initial value, $\underline{t}$, a lower bound for the optimal t, and then it updates $\underline{t}$ to $\mathrm{rbf}_i(\underline{t})$ repeatedly until $\underline{t}$ stabilizes or $\underline{t} > D_i$. The number of iterations is at most $D_i/\min_{j\in[i-1]} T_i$, and the algorithm has pseudo-polynomial running time. Of course, Γ is schedulable if and only if RTA finds feasible solutions for Problem (3) for all $i \in [n]$.

The next corollary follows directly from Theorem 1.1 and Theorem 2.1.

Corollary 2.2. *The collection of m servers at priority k is feasible if and only if for all $i \in [n] \setminus [k-1]$,*

$$\exists t \in (0, D_i] : \mathrm{rbf}_i(t) + \sum_{j\in[m]} \left\lceil \frac{t}{p_j} \right\rceil b_j \leq t.$$

3 MAXIMIZE BUDGET AND UTILIZATION SEPARATELY

Before we attempt to solve Dimensioning, we need to be able to find the maximum cumulative budget and the maximum cumulative utilization of the collection of servers separately.

[6] If $p_1 < B_{\min}$ then the servers can serve a task with execution requirement strictly greater than $\sum_j b_j$ immediately in some scenarios; we ignore this complication by assuming that $p_1 \geq B_{\min}$.
[7] Usually rbf is defined with respect to a task in Γ (see, for instance, [25, Sec. 3.3.3]) but it is more efficient for us to define it with respect to a subsystem of Γ.

Corollary 3.1. *Given Γ and k, let $B_{\max}$ denote the maximum cumulative budget for any feasible collection of servers at priority k. Then, we must have*

$$B_{\max} = \min_{i\in[n]\setminus[k-1]} \left\{ \max_{t\in(0,D_i]} \{t - \mathrm{rbf}_i(t)\} \right\}.$$

Proof. Since the servers are feasible, using Corollary 2.2, we must have

$$\forall i \in [n] \setminus [k-1] : \exists t \in (0, D_i] : \sum_{j\in[m]} \left\lceil \frac{t}{p_j} \right\rceil b_j \le t - \mathrm{rbf}_i(t)$$

Since the objective is to maximize $\sum_j b_j$, we can eliminate p_j from the above condition by choosing $p_j = \max_{i\in[n]\setminus[k-1]} D_i$ because this minimizes the coefficient of b_j. Then, the above condition simplifies to

$$\forall i \in [n] \setminus [k-1] : \exists t \in (0, D_i] : \sum_{j\in[m]} b_j \le t - \mathrm{rbf}_i(t)$$

The expression in the statement of the corollary is the tightest upper bound for $\sum_{j\in[m]} b_j$ in this condition. □

Corollary 3.2. *Given Γ and k, let $U_{\max}$ denote the maximum cumulative utilization for any feasible collection of servers at priority k. Then, we must have*

$$U_{\max} = \min_{i\in[n]\setminus[k-1]} \left\{ \max_{t\in(0,D_i]} \left\{ 1 - \frac{\mathrm{rbf}_i(t)}{t} \right\} \right\}.$$

Proof. Since the servers are feasible, using Corollary 2.2, we must have

$$\forall i \in [n] \setminus [k-1] : \exists t \in (0, D_i] : \sum_{j\in[m]} \left\lceil \frac{t}{p_j} \right\rceil p_j U_j \le t - \mathrm{rbf}_i(t)$$

Let us assume that an optimal solution exists where $t_i \in (0, D_i] \cap \mathbb{Q}$ satisfies the above inequality for each i. Since the objective is to maximize $\sum_j U_j$, we can eliminate p_j from the above condition by choosing $p_j = \gcd_{i\in[n]\setminus[k-1]} t_i$ because this minimizes the coefficient of U_j. Then, the above condition simplifies to

$$\forall i \in [n] \setminus [k-1] : \exists t \in (0, D_i] : \sum_{j\in[m]} U_j \le 1 - \frac{\mathrm{rbf}_i(t)}{t}$$

The expression in the statement of the corollary is the tightest upper bound for $\sum_{j\in[m]} b_j$ in this condition.

The global maximum point of $1 - \frac{\mathrm{rbf}_i(t)}{t}$ must be D_i or a multiple of a period T_j with $j \in [i]$, which are all rational values; thus, our assumption that there exist optimal solutions with rational t_i's is justified. □

From the above proofs, it is evident that a server with a large period such as $\max_{i\in[n]\setminus[k-1]} D_i$ can have budget $B_{\max}$ and a server with a small period $\gcd\{T_1, D_1, \ldots, T_n, D_n\}$ can have utilization $U_{\max}$. Better periods can be chosen for these solitary servers by analyzing the functions $t \mapsto t - \mathrm{rbf}_i(t)$ and $t \mapsto 1 - \mathrm{rbf}_i(t)/t$ in a little more detail. For any $i \in [n] \setminus [k-1]$, let β_i and μ_i be defined as follows:

$$\beta_i = \arg\max_{t\in(0,D_i]} \{t - \mathrm{rbf}_i(t)\} \tag{4}$$

$$\mu_i = \arg\max_{t\in(0,D_i]} \left\{ 1 - \frac{\mathrm{rbf}_i(t)}{t} \right\} \tag{5}$$

If multiple global maximum points are available, then we let β_i (resp., μ_i) denote the smallest global maximum point in the interval, thus ensuring that β_i and μ_i are well-defined. The following theorem follows from these definitions:

Theorem 3.3. *Given Γ and k, a server with budget $B_{\max}$ and period $\max_{i\in[n]\setminus[k-1]} \beta_i$ is feasible; and a server with utilization $U_{\max}$ and period $\gcd_{i\in[n]\setminus[k-1]} \mu_i$ is feasible.*

In the remainder of the section, we discuss algorithms for computing β_i, μ_i, $B_{\max}$, and $U_{\max}$.

Theorem 3.4. *Algorithm 1 computes*

$$\left(\max_{t\in(0,D_i]} \{t - \mathrm{rbf}_i(t)\}, \beta_i \right).$$

Proof Sketch. The correctness of the algorithm follows from three observations:

(i) RTA can be used to solve $\min\{t \in (0, D_i] \mid m + \mathrm{rbf}_i(t) \le t\}$ for a fixed m since the change is equivalent to modifying C_i to $C_i + m$.

(ii) The set

$$\{z \in (0, D_i] \mid \exists t \in (0, D_i] : z + \mathrm{rbf}_i(t) \le t\}$$

is a down-set[8], and, hence, binary search can be used to find the largest value in the set. Thus, we must have

$$x = \max_{t\in(0,D_i]} \{t - \mathrm{rbf}_i(t)\} \tag{6}$$

(iii) The z returned by the algorithm equals $\min\{t \in (0, D_i] \mid x + \mathrm{rbf}_i(t) \le t\}$. Thus, we must have

$$\begin{aligned}
& z - \mathrm{rbf}_i(z) = x \\
\implies & z - \mathrm{rbf}_i(z) = \max_{t\in(0,D_i]} \{t - \mathrm{rbf}_i(t)\} && \text{(using Equation (6))} \\
\implies & z = \arg\max_{t\in(0,D_i]} \{t - \mathrm{rbf}_i(t)\} \\
\implies & z = \beta_i && \text{(using Definition (4))}
\end{aligned}$$

□

Algorithm 2 is similar to Algorithm 1, and the next theorem can be proved using the same strategy as the above proof.

Theorem 3.5. *Algorithm 2 computes*

$$\left(\max_{t\in(0,D_i]} \left\{ 1 - \frac{\mathrm{rbf}_i(t)}{t} \right\}, \mu_i \right)$$

Algorithm 1 (resp., Algorithm 2) can be called for all $i \in [n] \setminus [k-1]$, and the minimum value amongst all values returned by the algorithm is $B_{\max}$ (resp., $U_{\max}$).

4 A TRACTABLE SPECIAL CASE OF DIMENSIONING

In this section, we solve Dimensioning in a restricted setting:

(i) The scheduler assigns rate-monotonic priorities to Γ_n, i.e.,

$$\forall i, j \in [n] : i \le j \implies T_i \le T_j. \tag{7}$$

[8]For an order P, a subset $A \subseteq P$ is a *down-set* if $x \in A$ and $y \le x$ imply that $y \in A$. We are implicitly working in the order $(\mathbb{Q}, \le)$ whenever we use the term down-set.

Algorithm 1 Compute $\left(\max_{t\in(0,D_i]}\{t-\mathrm{rbf}_i(t)\},\beta_i\right)$.

$(x,y,z)\leftarrow(0,D_i,0)$
repeat
  $m\leftarrow(x+y)/2$
  Use RTA to solve $\min\{t\in(0,D_i] \mid m+\mathrm{rbf}_i(t)\le t\}$.
  if an optimal solution exists **then**
    $x\leftarrow m$
    $z\leftarrow$ the optimal value
  else
    $y\leftarrow m$
  end if
until $x\approx y$
return (x,z)

Algorithm 2 Compute $\left(\max_{t\in(0,D_i]}\left\{1-\frac{\mathrm{rbf}_i(t)}{t}\right\},\mu_i\right)$.

$(x,y,z)\leftarrow(0,1,0)$
repeat
  $m\leftarrow(x+y)/2$
  Use RTA to solve $\min\{t\in(0,D_i] \mid \mathrm{rbf}_i(t)\le(1-m)t\}$.
  if an optimal solution exists **then**
    $x\leftarrow m$
    $z\leftarrow$ the optimal value
  else
    $y\leftarrow m$
  end if
until $x\approx y$
return (x,z)

(ii) The periods in Γ_n are harmonic, i.e.,

$$\forall i,j\in[n]: T_i \mid T_j \vee T_j \mid T_i. \tag{8}$$

(iii) The relative deadlines are all equal to their corresponding periods, i.e.,

$$\forall i,j\in[n]: D_i=T_i. \tag{9}$$

The first two assumptions imply that

$$\forall i,j\in[n]: i\le j \implies T_i \mid T_j. \tag{10}$$

For these restricted systems, it is not too hard to show that

$$\forall i,j\in[n]\setminus[k-1]: \beta_i=\mu_i=T_i \tag{11}$$

$$\forall i\in[n]: \mathrm{rbf}_i(T_i)=T_i\textstyle\sum_{j\in[i]}U_j \tag{12}$$

Then, using Corollary 3.1, Corollary 3.2, Definition (4) and Definition (5), we get the following identities:

$$B_{\max}=\min_{i\in[n]\setminus[k-1]}\left\{T_i(1-\textstyle\sum_{j\in[i]}U_j)\right\} \tag{13}$$

$$U_{\max}=1-\textstyle\sum_{j\in[n]}U_j \tag{14}$$

Thus, $B_{\max}$ and $U_{\max}$ can be computed efficiently for Γ_n. Let ℓ be defined as follows:

$$\ell=\max\{l\in[n]\setminus[k-1] \mid \beta_l-\mathrm{rbf}_l(\beta_l)=B_{\max}\} \tag{15}$$

Using Equations (11,12), ℓ is the largest number in $[n]\setminus[k-1]$ such that

$$T_\ell(1-\textstyle\sum_{j\in[\ell]}U_j)=B_{\max} \tag{16}$$

In the next theorem, we show that Dimensioning can be solved efficiently for systems if they satisfy the assumptions laid out at the beginning of the section.

Theorem 4.1. *If priorities are rate-monotonic, periods are harmonic, and deadlines are equal to periods, then for an instance* $(\Gamma_n, B_{\min}, k)$ *of Dimensioning exactly one of the following statements is true:*

(i) $B_{\max}$ *is less than* $B_{\min}$ *and hence the instance is infeasible.*
(ii) $B_{\max}$ *is at least* $B_{\min}$*. In this case, an optimal solution* $\langle(b_1,p_1),(b_2,p_2)\rangle$ *can be determined from the following properties:*

$$p_1=\max\{T_i \mid i\in[n]\setminus[\ell-1], T_i\le B_{\max}/U_{\max}\} \tag{17}$$

$$p_2=\min\{T_i \mid i\in[n]\setminus[\ell-1], T_i\ge B_{\max}/U_{\max}\} \tag{18}$$

$$b_1+b_2=B_{\max} \tag{19}$$

$$\frac{b_1}{p_1}+\frac{b_2}{p_2}=U_{\max} \tag{20}$$

$$b_1,b_2\ge 0 \tag{21}$$

Thus, the optimal objective value is $U_{\max}$*; the optimal solution and objective value are both independent of* $B_{\min}$*.*

Proof Sketch. We only consider the case where $B_{\max}$ is at least $B_{\min}$ because the other case is trivial.

First, we show that p_1 and p_2 are well-defined. From Equations (16,14), we get

$$B_{\max}<T_\ell\le\frac{B_{\max}}{U_{\max}} \tag{22}$$

Thus, we have

$$T_\ell\in\{T_i \mid i\in[n]\setminus[\ell-1], T_i\le B_{\max}/U_{\max}\}\neq\emptyset,$$

and hence p_1 is well-defined. Similarly, by using Equations (13,14), we can show that $T_n\ge B_{\max}/U_{\max}$ and hence p_2 is well-defined. We note that p_1 must be equal to T_α for some $\alpha\in[n]\setminus[\ell-1]$, and p_2 must be equal to T_α or $T_{\min(\alpha+1,n)}$. Thus, we must have

$$T_\ell\le T_\alpha=p_1\le p_2\le T_{\min(\alpha+1,n)}\le T_n \tag{23}$$

Next, we show that we can always find values for b_1 and b_2 that satisfy Equations (19–21). If p_1 and p_2 are equal, then $(b_1,b_2)=(B_{\max},0)$ works; otherwise, from Equations (19,20), we get

$$b_1=\frac{U_{\max}-\frac{B_{\max}}{p_2}}{\frac{1}{p_1}-\frac{1}{p_2}}$$

$$b_2=\frac{\frac{B_{\max}}{p_1}-U_{\max}}{\frac{1}{p_1}-\frac{1}{p_2}}$$

Then, from Equations (17,18,23), it follows that b_1 and b_2 are nonnegative.

The first two constraints in Dimensioning are met by b_1, p_1, b_2, and p_2 because

$$B_{\min}\le b_1+b_2=B_{\max}<T_\ell\le p_1\le p_2$$

We assumed $B_{\min}\le B_{\max}$ when we started the proof; the second inequality follows from Equation (16), and the other inequalities follow from Equation (23). Equation (20) implies that the objective of maximizing the utilization is achieved by $\langle(b_1,p_1),(b_2,p_2)\rangle$. Thus,

we can complete the proof by verifying that the servers are feasible, i.e., they satisfy the third constraint in Dimensioning. Using Corollary 2.2, the server is feasible if and only if

$$\forall i \in [n]\backslash[k-1] : \exists t \in (0, D_i] : \text{rbf}_i(t) + \left\lceil \frac{t}{p_1} \right\rceil b_1 + \left\lceil \frac{t}{p_2} \right\rceil b_2 \leq t \quad (24)$$

We will prove the above statement by considering two cases in the next two paragraphs.

Case I. For any $i \in [\alpha] - [k-1]$, we can choose $t = T_i$ to get

$$\begin{aligned}
&\text{rbf}_i(T_i) + \left\lceil \frac{T_i}{p_1} \right\rceil b_1 + \left\lceil \frac{T_i}{p_2} \right\rceil b_2 \\
=&\text{rbf}_i(T_i) + b_1 + b_2 && \text{(using Equation (23), Assumption (7))} \\
=&T_i \textstyle\sum_{j\in[i]} U_j + b_1 + b_2 && \text{(using Equation (12))} \\
=&T_i \textstyle\sum_{j\in[i]} U_j + B_{\max} && \text{(using Equation (19))} \\
=&T_i + B_{\max} - T_i(1 - \textstyle\sum_{j\in[i]} U_j) \\
\leq&T_i && \text{(using Equation (13))}
\end{aligned}$$

Case II. For any $i \in [n] \setminus [\alpha]$, we can choose $t = T_i$ to get

$$\begin{aligned}
&\text{rbf}_i(T_i) + \left\lceil \frac{T_i}{p_1} \right\rceil b_1 + \left\lceil \frac{T_i}{p_2} \right\rceil b_2 \\
=&\text{rbf}_i(T_i) + \frac{T_i}{p_1} b_1 + \frac{T_i}{p_2} b_2 \\
& && \text{(using Equation (23), Assumption (10))} \\
=&\text{rbf}_i(T_i) + T_i U_{\max} && \text{(using Equation (20))} \\
=&T_i \textstyle\sum_{j\in[i]} U_j + T_i U_{\max} && \text{(using Equation (12))} \\
=&T_i + T_i(U_{\max} - (1 - \textstyle\sum_{j\in[i]} U_j)) \\
\leq&T_i && \text{(using Equation (14))}
\end{aligned}$$

□

We distill the analysis carried out in this section into Algorithm 3 which solves Dimensioning in the special case studied in this section. Lines 1,5–8 run in $O(n)$ time; the remaining lines run in $O(1)$ time. The next theorem follows.

Theorem 4.2. *When priorities are rate-monotonic, periods are harmonic, and deadlines are equal to periods, Algorithm 3 solves Dimensioning in $O(n)$ time.*

We examine the behavior of Algorithm 3 by varying a parameter of a simple system in the next example.

Example 4.3. Table 1 shows the solution to Dimensioning found by Algorithm 3 for four systems when $k = 1$ and $B_{\min} \leq B_{\max}$. The systems differ only in the wcet of their lowest priority task τ_3. Each server has a budget of 4 units and consumes all the residual utilization, i.e., $1 - \sum_{i\in[3]} U_i$, in the system. As we go down the table, the residual utilization in the system decreases because C_3 increases; the last two columns show that Algorithm 3 allocates more budget to larger periods as we go down the table.

5 DIMENSIONING IN GENERAL

We formulate Dimensioning for general FP systems as an MINLP (mixed-integer nonlinear program) in Figure 1. The formulation is not exact yet because we treat m as a constant in the program and Dimensioning requires m to be arbitrary (we will correct this

Algorithm 3 Solve Dimensioning in special case.

1: $B_{\max} \leftarrow \min_{i\in[n]\setminus[k-1]} \{T_i(1 - \sum_{j\in[i]} U_j)\}$
2: **if** $B_{\max} < B_{\min}$ **then**
3: **return** "infeasible"
4: **end if**
5: $\ell \leftarrow \max\{l \in [n] \setminus [k-1] \mid T_l(1 - \sum_{j\in[l]} U_j) = B_{\max}\}$
6: $U_{\max} \leftarrow 1 - \sum_{j\in[n]} U_j$
7: $p_1 \leftarrow \max\{T_i \mid i \in [n] \setminus [\ell - 1], T_i \leq B_{\max}/U_{\max}\}$
8: $p_2 \leftarrow \min\{T_i \mid i \in [n] \setminus [\ell - 1], T_i \geq B_{\max}/U_{\max}\}$
9: **if** $p_1 = p_2$ **then**
10: **return** $\langle (B_{\max}, p_1) \rangle$
11: **end if**
12: $x \leftarrow U_{\max} - B_{\max}/p_2$
13: $y \leftarrow 1/p_1 - 1/p_2$
14: $b_1 \leftarrow x/y$
15: $b_2 \leftarrow B_{\max} - b_1$
16: **return** $\langle (b_1, p_1), (b_2, p_2) \rangle$

Table 1: Four Dimensioning instances solved by Algorithm 3

Γ	Optimal solution	Periods in solution
$\langle (1,5), (3,10), (0,20) \rangle$	$\langle (1,5), (3,10) \rangle$	$\{5, 10\}$
$\langle (1,5), (3,10), (1,20) \rangle$	$\langle (0.5,5), (3.5,10) \rangle$	$\{5, 10\}$
$\langle (1,5), (3,10), (2,20) \rangle$	$\langle (4,10) \rangle$	$\{10\}$
$\langle (1,5), (3,10), (3,20) \rangle$	$\langle (3,10), (1,20) \rangle$	$\{10, 20\}$

departure from Dimensioning shortly). We observe the following facts about the program:

(i) The constants C, T, and D, which are vectors of dimension n, describe the FP subsystem Γ. The constants C, T, D, $B_{\min}$ and k describe an instance of Dimensioning.
(ii) The constant $B_{\max}$ (resp., $U_{\max}$) is computed for Γ and k using Algorithm 1 (resp., Algorithm 2).
(iii) The variables t, b, p, x and y are described in constraints 10–13. b and p are vectors of dimension m; we emphasize that m is fixed in the program. k, b and p describe the collection of servers. We will build up to an interpretation of t, x, and y in the following points.
(iv) The program and Dimensioning have the same objective.
(v) Constraint 1 is the first constraint in Dimensioning.
(vi) The nonnegativity of t, p, and P and constraints 5–6 imply that x and y are nonnegative.
(vii) The nonnegativity of x and y, the positivity of C, and constraint 4 imply that t is positive.
(viii) The positivity of t and constraint 6 imply that p is positive; thus, the objective function is well-defined.
(ix) The positivity of p and constraints 2–3 are collectively equivalent to the second constraint in Dimensioning.
(x) Since t_i is positive and at most D_i (constraint 7), we have $t_i \in (0, D_i]$.
(xi) In any optimal solution for the program, we must have $x_{i,j} = \lceil t_i/T_j \rceil$ and $y_{i,j} = \lceil t_i/p_j \rceil$ to ensure that b_j is maximal in constraint 4. Thus, in any optimal solution, for all $i \in [n] \setminus$

$[k-1]$, we must have

$$C_i + \sum_{j\in[i-1]} \lceil t_i/T_j\rceil C_j + \sum_{j\in[m]} \lceil t_i/p_j\rceil b_j \le t_i$$
$$\iff \sum_{j\in[i]} \lceil t_i/T_j\rceil C_j + \sum_{j\in[m]} \lceil t_i/p_j\rceil b_j \le t_i \quad (\text{since } t_i \in (0, D_i])$$
$$\iff \mathrm{rbf}_i(t_i) + \sum_{j\in[m]} \lceil t_i/p_j\rceil b_j \le t_i$$

Using Corollary 2.2, the above constraint is equivalent to the third constraint in Dimensioning.

(xii) Constraints 8-9 are not essential to the formulation but they help to strengthen it.

Therefore, the program is a formulation of Dimensioning for a fixed m. In the next theorems, we show that m can be restricted to $n-(k-1)$ in Dimensioning and therefore the MINLP with $m = n-(k-1)$ is an exact formulation of Dimensioning.

Theorem 5.1. *There exists an optimal solution to Dimensioning where $m \le n-(k-1)$.*

Proof. Consider an optimal collection of m servers such that $m > n-(k-1)$. This collection must satisfy the constraints in the above MINLP for some t', x', y', b', p'. In particular, constraint 4 must be satisfied:

$$\forall i \in [n] \setminus [k-1] : C_i + \sum_{j\in[i-1]} x'_{i,j} C_j + \sum_{j\in[m]} y'_{i,j} b'_j \le t'_i$$

Consider the following linear program where t', x', y', p' and C are constants and b is variable:

$$\begin{array}{lrcll} \max & \sum_{j\in[m]} b_j/p'_j & & & \\ \text{s.t.} & \sum_{j\in[m]} y'_{i,j} b_j & \le & t'_i - \sum_{j\in[i]} x'_{i,j} C_j, & i \in [n]\setminus[k-1] \\ & b_i & \ge & 0, & i \in [m] \\ & b & \in & \mathbb{R}^m & \end{array}$$

Clearly, b' is an optimal solution for the above program. Therefore, the program has basic feasible solutions. In any basic feasible solution, at least m constraints must be satisfied as equalities because the program has m variables. Since the program has $n-(k-1)$ constraints in the first line and m nonnegativity constraints in the second line, at least $m-n+(k-1)$ of the variables must be equal to zero in a basic feasible solution. Equivalently, at most $n-(k-1)$ variables are nonzero in a basic feasible solution. The nonzero variables and their corresponding periods in any basic feasible solution constitute an optimal solution to Dimensioning in which $m \le n-(k-1)$. □

The next theorem follows from the previous theorem and the observations made at the start of this section about the MINLP in Figure 1.

Theorem 5.2. *If m is equal to $n-(k-1)$ and $B_{\max}$ (resp., $U_{\max}$) has been computed using Algorithm 1 (resp., Algorithm 2), then the MINLP in Figure 1 is a valid formulation of Dimensioning.*

MINLPs can often be solved more efficiently when they are convex [6]. We note that constraints 4 and 6 are nonconvex; however, the constraints are quadratic and specialized approaches have been proposed for solving mixed-integer quadratically-constrained programs (MIQCPs) (see, for instance, [5]). We do not believe that simply feeding the MINLP in Figure 1 into various MINLP or MIQCP solvers and choosing the best solver based on these results is a good idea. There is a lot of structure in Dimensioning, and we are in the early stages of determining how Dimensioning can be decomposed into smaller problems that are tractable as MINLP formulations or otherwise; this structure is examined to a limited extent in the next subsection.

Recall from Theorem 3.3 that we one server with a small period such as $g = \gcd_{i\in[n]\setminus[k-1]} \mu_i$ and utilization $U_{\max}$ is feasible. Thus, if we get a Dimensioning instance $(\Gamma, k, B_{\min})$ where $B_{\min} \le gU_{\max}$, then the optimal value for the instance is $U_{\max}$. This idea that a Dimensioning instance with a *small* $B_{\min}$ can be solved immediately can be further strengthened by considering the restricted dual of Dimensioning described in the next subsection.

5.1 A dual of Dimensioning

In the dual problem of Dimensioning, the objective is to find $b_1, p_1, \ldots, b_m, p_m$ for an arbitrary m that maximize

$$\sum_{j\in[m]} b_j$$

subject to the following constraints:

(i) $\sum_{j\in[m]} b_j/p_j = U_{\max}$,
(ii) $\sum_{j\in[m]} b_j \le p_1 \le p_2 \le \cdots \le p_m$, and
(iii) the collection of servers $(k, b_1, p_1), (k, b_2, p_2), \ldots, (k, b_m, p_m)$ are feasible.

We denote the optimal budget for this problem by $B_{\max}(U_{\max})$. An instance of this problem is given by the pair (Γ, k). Any Dimensioning instance $(\Gamma, k, B_{\min})$ induces an instance (Γ, k) of the above problem; if the optimal value of the induced instance is at least $B_{\min}$ then the optimal solution of the induced instance is also an optimal solution of the original instance and the optimal value of the original instance is $U_{\max}$. Thus, the problem of computing $B_{\max}(U_{\max})$ is closely related to Dimensioning.

We will propose an MINLP for computing $B_{\max}(U_{\max})$ which is similar to the MINLP for Dimensioning (Figure 1); the major difference from the previous MINLP is that the periods are constant in this MINLP, and hence there are fewer nonlinear constraints. The next theorem explains why we can restrict our attention to a small set of periods.

Theorem 5.3. *For any collection of servers with cumulative utilization $U_{\max}$, the period of each server must divide*

$$g = \gcd_{i\in I} \mu_i, \tag{25}$$

where I is given by

$$I = \{i \in [n] \setminus [k-1] \mid 1 - \mathrm{rbf}_i(\mu_i)/\mu_i = U_{\max}\} \tag{26}$$

Proof Sketch. Let $i \in I$, and let the server have parameters $k, b_1, p_1, \ldots, b_m, p_m$. Recall from the proof of Corollary 3.2 that since the server is feasible we must have

$$\exists t \in (0, D_i] : \sum_{j\in[m]} \left\lceil \frac{t}{p_j} \right\rceil p_j U_j \le t - \mathrm{rbf}_i(t)$$

The lhs in the inequality is at least $U_{\max}$ and the rhs is at most $U_{\max}$ using Corollary 3.2 and Equations (5,26). Thus, the inequality must

$$
\begin{array}{lllll}
\max & \sum_{j\in[m]} b_j/p_j & & & \\
\text{s.t.} & (1) & \sum_{j\in[m]} b_j \geq B_{\min} & & \\
& (2) & \sum_{j\in[m]} b_j \leq p_1 & & \\
& (3) & p_i \leq p_j, & & i \in [m], j \in [i-1] \\
& (4) & C_i + \sum_{j\in[i-1]} x_{i,j} C_j + \sum_{j\in[m]} y_{i,j} b_j \leq t_i, & & i \in [n] \setminus [k-1] \\
& (5) & t_i \leq T_j x_{i,j}, & & i \in [n] \setminus [k-1], j \in [i-1] \\
& (6) & t_i \leq p_j y_{i,j}, & & i \in [n] \setminus [k-1], j \in [m] \\
& (7) & t_i \leq D_i, & & i \in [n] \setminus [k-1] \\
& (8) & \sum_{j\in[m]} b_j \leq B_{\max} & & \\
& (9) & \sum_{j\in[m]} b_j/p_j \leq U_{\max} & & \\
& (10) & b, p \in \mathbb{R}^m_{\geq 0} & & \\
& (11) & t \in \mathbb{R}^{n-(k-1)}_{\geq 0} & & \\
& (12) & x \in \mathbb{Z}^{(n(n-1)-(k-1)(k-2))/2} & & \\
& (13) & y \in \mathbb{Z}^{(n-(k-1))m} & &
\end{array}
$$

Figure 1: MINLP for Dimensioning

be satisfied as an equality with $t = \mu_i$:

$$\sum_{j\in[m]} \left\lceil \frac{\mu_i}{p_j} \right\rceil p_j U_j = U_{\max}$$

p_j must divide μ_i to ensure that $\sum_j U_j = U_{\max}$. Thus, p_j divides μ_i for all $i \in I$, and hence it divides g. □

Let g' be given by

$$g' = \gcd_{i\in[n]\setminus[k-1]} \mu_i \tag{27}$$

From Theorem 3.3, we know that $B_{\max}(U_{\max})$ is at least $U_{\max} g'$. Thus, $U_{\max} g'$ is a lower bound for $\sum_j b_j$, which in turn is a lower bound for the periods, using the second constraint in the problem. Thus, periods can be chosen from the following set:

$$\{g/x \mid x \in \mathbb{N}_{>0}, g/x > U_{\max} g'\}$$

$U_{\max} g$ is the budget corresponding to a single server with period g. We can perform binary search in the above set to find the largest period g'' such that the server $(k, U_{\max} g'', g'')$ is feasible. $U_{\max} g''$ is the largest budget for one server with utilization $U_{\max}$ and we denote it as $B^1_{\max}(U_{\max})$. Now, periods can be chosen from the set

$$\{g/x \mid x \in \mathbb{N}_{>0}, g/x > B^1_{\max}(U_{\max})\}.$$

The MINLP in Figure 1 can be modified as follows to solve the current problem:

(i) p is now a vector constant with elements

$$\{g/x \mid x \in \mathbb{N}_{>0}, g/x > B^1_{\max}(U_{\max})\}.$$

in increasing order. Like the previous MINLP, m still denotes the length of p, or equivalently the length of b. Unlike the previous, m is not equal to $n-(k-1)$; instead, it is equal to $|p|$ since p is constant in this MINLP.

(ii) The objective is $\max \sum_{j\in[m]} b_j$.

(iii) Constraints 1–3 are replaced by the following constraints

$$
\begin{array}{rcll}
\sum_{j\in[m]} b_j & \geq & B_1(U_{\max}) & \\
\sum_{j\in[m]} b_j/p_j & = & U_{\max} & \\
b_i > 0 & \implies & \sum_{j\in[m]} b_j \leq p_i, & i \in [m]
\end{array}
$$

(iv) In constraints 4–7, i can be restricted to $([n] \setminus [k-1]) \setminus I$ because when $i \in I$ the constraints are collectively equivalent to the utilization constraint included above, i.e.,

$$\sum_{j\in[m]} b_j/p_j = U_{\max}.$$

We note that constraint 6 is a linear constraint in this program (it was nonlinear in the MINLP for Dimensioning). Reducing nonlinearity in an MINLP brings it closer to a MILP making it more tractable. However, we are not able to eliminate nonlinearity since constraint 4 is still quadratic.

5.2 An example demonstrating the use of the two MINLPs

Consider the following Dimensioning instance $\Gamma = \langle (1,4), (1,7) \rangle$, $k = 1$, and an arbitrary $B_{\min}$. Using Algorithm 1, we get the following values for β_i:

i	β_i	$\beta_i - \text{rbf}_i(\beta_i)$
1	4	3
2	7	4

From Corollary 3.1, $B_{\max}$ is equal to 3. Using Algorithm 2, we get the following values for μ_i:

i	μ_i	$1 - \text{rbf}_i(\mu_i)/\mu_i$
1	4	3/4
2	7	4/7

From Corollary 3.2, $U_{\max}$ is equal to $4/7 \approx 0.5714$.

First, we solve the dual of Dimensioning. From Definitions (26, 25, 27), it follows that $I = \{2\}$, $g = \gcd(7) = 7$, and $g' = \gcd(4, 7) = 1$. Thus, periods can be chosen from the set

$$\{7/x \mid x \in \mathbb{N}_{>0}, 7/x > 4/7\} = \{7, 7/2, \ldots, 7/12\}$$

By performing binary search on this set, we find that amongst all servers with utilization 4/7, the server $(2, 7/2)$ has the largest period, i.e., $B^1_{\max}(U_{\max}) = 2$. Thus, the periods can be restricted even further to the set

$$\{7/x \mid x \in \mathbb{N}_{>0}, 7/x > 2\} = \{7, 7/2, 7/3\}$$

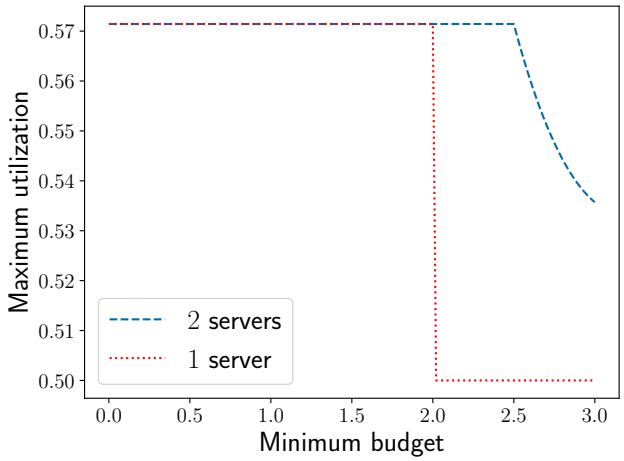

Figure 2: The dashed blue line shows optimal values of Dimensioning where $\Gamma = \langle(1,4),(1,7)\rangle$, $k = 1$, and $B_{\min} \in [0, B_{\max}]$; the dotted red line shows optimal values when only one server is used.

We use the global optimization solver BARON [27] to solve the MINLP for the dual. The solver returns $\langle(3/2, 7/2),(1,7)\rangle$ as an optimal solution, and the optimal value $B_{\max}(U_{\max})$ is equal to $5/2$.

For any instance of Dimensioning with $B_{\min} \leq 5/2$ the optimal value is $U_{\max} \approx 0.5714$. To understand how the optimal value varies as a function of $B_{\min}$ in $(2.5, 3]$ (recall that $B_{\max} = 3$), we solve 10 instances of the MINLP for Dimensioning where $B_{\min} \in \{2.55, 2.60, \ldots, 3.00\}$ and $m = 2$ (recall from Theorem 5.1 that the choice $m = n - (k-1)$ can be made without any loss of generality). The optimal solution in all 10 cases is given by

$$\langle(4 - B_{\min}, B_{\min} + 1), (2B_{\min} - 4, 7)\rangle,$$

and the optimal value is

$$\frac{24 - 9B_{\min} + 2B_{\min}^2}{7B_{\min} + 7}$$

We show the results in Figure 2. While the maximum utilization for one server sharply drops off at $B_{\min} = 2$ from $4/7$ to $1/2$, the maximum utilization for m servers remains at $4/7$ until $B_{\min} = 2.5$ and then it gradually decreases to $15/28$. Thus, multiple servers help in finding a better middle ground for the conflicting objectives of maximizing budget and maximizing utilization than one server.

The methodology that we have described in this section can be repeated for any (Γ, k), in principle, to visualize the full design space for Dimensioning; however, more work needs to be done to understand how to make it scale for larger Γ.

6 IMPLEMENTATION ISSUES

When implementing m servers at the same priority k, we have two choices:

(i) Implement a server which simulates a single task. For each $i \in [m]$, instantiate a server which simulates a regular task (b_i, p_i).

(ii) Implement a server which simulates multiple tasks, and instantiate it once.

In the first case, when a server's budget is exhausted and the current job is still pending, we need to switch over to a different server which has nonzero budget; the time used in switching contexts is wasteful. In the second case, some needless context switches can be avoided, but we must implement a server which can simulate multiple tasks. To the best of our knowledge, such servers have not been specified before. Now, we will outline the specifications of a regular sporadic server which is capable of simulating a single task and an extended sporadic server which is capable of simulating multiple tasks. This extension can also be used as a blueprint for extending aperiodic servers other than the sporadic server.

6.1 A Regular Sporadic Server

The sporadic server simulates a *swarm* of small sporadic tasks with the same period P and cumulative wcet B [24]. From a schedulability perspective, the sporadic server simulates a single task (B, P) since there is no distinction between the swarm and the single task at a *critical instant* [17]. The following description of a sporadic server is obtained by simplifying a corrected `POSIX` sporadic server specification [24, Sec. 4].

At any instant, the server maintains a nonempty queue,

$$q = \langle(a_1, b_1), (a_2, b_2), \ldots, (a_x, b_x)\rangle.$$

Each element (a_i, b_i) denotes a chunk b_i of the budget B which becomes *active* at time a_i. The server ensures that q is sorted in nondecreasing order of activation times, and that the sum of the chunks of budget in q equals B; thus, there is no loss or amplification of budget.

The *formal budget* at an instant t is 0 if the first chunk in q is not active, i.e., $a_1 > t$; otherwise, the formal budget at t is $b_1 - usg$, where usg is a variable maintained by the server which stores the portion of b_1 that has been consumed by the server. A server is said to be *exhausted* at an instant t if the formal budget at t is less than or equal to zero. The formal budget at t is a lower bound for the *true budget* at t because a second chunk in q might become active before exhaustion occurs at $t + b_1 - usg$ if $a_2 \leq t + b_1 - usg$. If $a_2 \leq t + b_1 - usg$, then the true budget is at least $b_1 + b_2 - usg$. Thus, the true budget is equal to the sum of the budgets of chunks with indices in the set given by

$$\{i \in [x] \mid a_i \leq t + \textstyle\sum_{j \in [i-1]} b_i - usg\} \tag{28}$$

At certain points in the execution, the server syncs the formal budget and the true budget by merging chunks that contribute to the true budget into one chunk.

Initially, q equals $\langle(0, B)\rangle$ and usg equals 0. The server modifies its state (q, usg) in the following cases:

(i) *An aperiodic request arrives at instant t when there are no pending aperiodic requests and $a_1 \leq t$.* All chunks that contribute to the true budget are merged into a single chunk with activation time t, and then the formal budget equals the true budget. We demonstrate this rule through an example: say that when the request arrives at time t, we have

$$q = \langle(t-5, 1), (t-3, 1), (t+2, 1)\rangle, usg = 0.$$

All three chunks can be used to serve aperiodic requests continuously in the interval $[t, t+2]$. Therefore, we merge the chunks into one, and we have

$$q = \langle (t, 3) \rangle, usg = 0.$$

(ii) *The server begins execution at t_1 and ends at t_2.* The server stops executing due to preemption, exhaustion, or completion of all pending aperiodic tasks. At t_2, usg is increased by $t_2 - t_1$:

$$[usg]_{\text{at } t_2} = [usg]_{\text{at } t_1} + t_2 - t_1$$

q, and b_1 in particular, do not change between t_1 and t_2. Thus, the formal budget at t_2 is $t_2 - t_1$ less than the formal budget at t_1; the server ensures that

$$t_2 - t_1 \leq [b_1 - usg]_{\text{at } t_1},$$

thus ensuring that the formal budget at t_2 is nonnegative.

(iii) *At instant t, the server stops executing due to exhaustion.* A new element $(a_1 + P, b_1)$ is pushed to the back of q, the head of q is popped off, and usg is reset to 0.

(iv) *At instant t, the server stops executing due to completion of all pending requests but it is not exhausted.* A new element $(a_1 + P, usg)$ is pushed to the back of the queue, the head of q is modified to $(a_1, b_1 - usg)$, and usg is reset to 0. We note that $b_1 - usg > 0$ because the server is not exhausted.

6.2 An Extended Sporadic Server

The extended sporadic server simulates m swarms of small sporadic tasks where the i-th swarm contains tasks with the same period P_i and cumulative wcet B_i. From a schedulability perspective, the extended sporadic server simulates

$$\langle (B_1, P_1), (B_2, P_2), \ldots, (B_m, P_m) \rangle.$$

We assume that $P_1 < P_2 < \cdots < P_m$.

At any instant, the server maintains a nonempty queue,

$$q = \langle (a_1, b_1, c_1), (a_2, b_2, c_2), \ldots, (a_x, b_x, c_x) \rangle.$$

Each element (a_i, b_i, c_i) denotes a chunk b_i of the budget associated with period $c_i \in \{P_1, \ldots, P_m\}$; a_i is the time at which the chunk becomes active. The server ensures that for any two indices i, j we have

$$i < j \implies a_i < a_j \vee (a_i = a_j \wedge c_i < c_j) \tag{29}$$

Thus, the elements are sorted primarily in nondecreasing order of activation times, and secondarily in rate-monotonic order. The server also ensures that the sum of the chunks of budget in q with period P_i is equal to B_i for each $i \in [m]$; thus, there is no loss or amplification of the budget associated with any period.

For the regular sporadic server, the formal budget is synced with the true budget by merging adjacent chunks but in the extended sporadic server adjacent chunks in q may have different periods, not allowing them to be merged with each other. We use a new variable f to store the cumulative budget of one or more chunks at the head of q. The formal budget at an instant t is 0 if $a_1 > t$; otherwise, the formal budget at t is $f - usg$, where usg stores the portion of f that has been consumed by the server.

Initially, q equals

$$\langle (0, B_1, P_1), (0, B_2, P_2), \ldots, (0, B_m, P_m) \rangle,$$

f equals $\sum_{j \in [m]} B_j$, and usg equals 0. The server modifies its state (q, f, usg) in the following cases:

(i) *An aperiodic request arrives at instant t when there are no pending aperiodic requests and $a_1 \leq t$.* If a chunk at index i contributes to the true budget, then a_i is changed to $t + \sum_{j \in [i-1]} b_i - usg$. All adjacent chunks with the same period that contribute to the true budget are merged into a single chunk. f is set to the true budget. We demonstrate this rule using an example: say that when the request arrives at time t, we have

$$q = \langle (t-5, 1, 5), (t-3, 1, 3), (t+2, 1, 3) \rangle, f = 0, usg = 0.$$

All three chunks can be used to serve aperiodic requests continuously in the interval $[t, t+2]$ and the last two chunks can be merged because they have the same period. We modify q and f to get

$$q = \langle (t, 1, 5), (t+1, 2, 3) \rangle, f = 3, usg = 0.$$

The astute reader will observe that instead of the above modification we could also have reordered q safely to get

$$q = \langle (t, 1, 3), (t, 1, 5), (t+2, 1, 3) \rangle, f = 3, usg = 0.$$

The reordering allows q to be more rate-monotonic and leads to smaller activation times in a behavior of the server; however, reordering also requires more computation compared to our initial proposal, which can be executed in linear time.

(ii) *The server begins execution at t_1 and ends at t_2.* As before, usg must be increased by $t_2 - t_1$, and the server ensures that the formal budget at t_2 is nonnegative by limiting the duration of the execution:

$$t_2 - t_1 \leq [f - usg]_{\text{at } t_1}$$

(iii) *At instant t, the server stops executing due to exhaustion.* The cumulative budget of a number of chunks at the head of q, stored in f, has been completely consumed; thus, now we have $f = usg$. The head of q is popped off, a new element $(a_1 + c_1, b_1, c_1)$ is inserted in q at a position that maintains the order described in Equation (29), and usg is decreased by b_1; this process is repeated until $usg = 0$. Finally, f is reset to the true budget.

(iv) *At instant t, the server stops executing due to completion of all pending requests but it is not exhausted.* The process described in the previous step is also repeated here until we have $usg < b_1$. Then, the head of q is modified to $(a_1, b_1 - usg, c_1)$, a new element $(a_1 + c_1, usg, c_1)$ is inserted in q at a position that maintains the order described in Equation (29), usg is reset to 0, and f is reset to the true budget.

We note that the extended server reduces the unnecessary context switch overhead mentioned at the start of Section 6 by keeping track of the cumulative budget associated with different periods through the variable f. However, maintaining q and f takes more time and q takes more space since it stores triples. We have presented an informal specification for the extended sporadic server but its effectiveness for specific applications and hardware platforms has not been addressed here.

7 CONCLUSION

We proposed Dimensioning, an optimization problem which allows a system designer to understand the trade-off between choosing a large budget and a large utilization when multiple servers are allowed to run at a priority of their choice. In the restricted case of harmonic rate-monotonic implicit-deadline FP systems, we showed that the trade-off is nonexistent because a collection of two servers with budget $B_{\max}$ and utilization $U_{\max}$ can be found in linear time (Algorithm 3). We showed that the trade-off is present in general systems; for instance, in Figure 2 if the servers must have a minimum budget of 2.2 (resp., 3) then their maximum cumulative utilization is 0.571 (resp., 0.535). We showed how to compute the right end of the flat part of the trade-off curve using one MINLP (Section 5.1), and we showed how to compute points on the remainder of the trade-off curve using another MINLP (Figure 1). We proposed a new specification for an extended sporadic server which can simulate multiple tasks at the same priority level (Section 6).

Many ideas introduced in this work need further investigation; for instance, we do not know the answers to the following questions:

(1) Can we efficiently solve more general cases than the case described in Section 4?
(2) How well do the MINLPs proposed in Section 5 scale for larger instances of Dimensioning? Which MINLP algorithms are most effective for instances generated in practice? Can we solve the general case without using MINLPs?
(3) Is the extended sporadic server specification in Section 6 an improvement over multiple regular sporadic servers for real applications?

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is supported by the US National Science Foundation under Grant numbers CPS-1932530, CNS-2141256 and CNS-2229290.

REFERENCES

[1] 2008. IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R)). *IEEE Std 1003.1-2008 (Revision of IEEE Std 1003.1-2004)* (Dec. 2008), 1–3874. https://doi.org/10.1109/IEEESTD.2008.4694976
[2] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. 1993. Applying New Scheduling Theory to Static Priority Pre-Emptive Scheduling. *Software Engineering Journal* 8, 5 (Sept. 1993), 284–292. https://doi.org/10.1049/sej.1993.0034
[3] Neil C. Audsley, Alan Burns, Robert I. Davis, Ken W. Tindell, and Andy J. Wellings. 1995. Fixed Priority Pre-Emptive Scheduling: An Historical Perspective. *Real-Time Systems* 8, 2-3 (1995), 173–198. https://doi.org/10.1007/BF01094342
[4] G. Bernat and A. Burns. 1999. New Results on Fixed Priority Aperiodic Servers. In *Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054)*. 68–78. https://doi.org/10.1109/REAL.1999.818829
[5] Timo Berthold, Stefan Heinz, and Stefan Vigerske. 2012. Extending a CIP Framework to Solve MIQCPs. In *Mixed Integer Nonlinear Programming (The IMA Volumes in Mathematics and Its Applications)*, Jon Lee and Sven Leyffer (Eds.). Springer, New York, NY, 427–444. https://doi.org/10.1007/978-1-4614-1927-3_15
[6] Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. 2012. Algorithms and Software for Convex Mixed Integer Nonlinear Programs. In *Mixed Integer Nonlinear Programming (The IMA Volumes in Mathematics and Its Applications)*, Jon Lee and Sven Leyffer (Eds.). Springer, New York, NY, 1–39. https://doi.org/10.1007/978-1-4614-1927-3_1
[7] Pieter J. L. Cuijpers and Reinder J. Bril. 2007. Towards Budgeting in Real-Time Calculus: Deferrable Servers. In *Formal Modeling and Analysis of Timed Systems (Lecture Notes in Computer Science)*, Jean-François Raskin and P. S. Thiagarajan (Eds.). Springer, Berlin, Heidelberg, 98–113. https://doi.org/10.1007/978-3-540-75454-1_9
[8] R.I. Davis and A. Burns. 2005. Hierarchical Fixed Priority Pre-Emptive Scheduling. In *26th IEEE International Real-Time Systems Symposium (RTSS'05)*. 10 pp.–398. https://doi.org/10.1109/RTSS.2005.25
[9] Rob Davis and Alan Burns. 2008. An Investigation into Server Parameter Selection for Hierarchical Fixed Priority Pre-emptive Systems. In *16th International Conference on Real-Time and Network Systems (RTNS 2008)*.
[10] M. Joseph and P. Pandya. 1986. Finding Response Times in a Real-Time System. *Comput. J.* 29, 5 (Jan. 1986), 390–395. https://doi.org/10.1093/comjnl/29.5.390
[11] M.H. Klein, J.P. Lehoczky, and R. Rajkumar. 1994. Rate-Monotonic Analysis for Real-Time Industrial Computing. *Computer* 27, 1 (Jan. 1994), 24–33. https://doi.org/10.1109/2.248876
[12] Mark H. Klein, Thomas Ralya, Bill Pollak, Ray Obenza, and Michael González Harbour. 1993. *A Practitioner's Handbook for Real-Time Analysis.* Kluwer Academic Publishers, USA.
[13] Tei-Wei Kuo and Ching-Hui Li. 1999. A Fixed-Priority-Driven Open Environment for Real-Time Applications. In *Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054)*. 256–267. https://doi.org/10.1109/REAL.1999.818851
[14] J. Lehoczky, L. Sha, and Y. Ding. 1989. The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. In *[1989] Proceedings. Real-Time Systems Symposium*. 166–171. https://doi.org/10.1109/REAL.1989.63567
[15] John P. Lehoczky, Lui Sha, and Jay K. Strosnider. 1987. Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. In *Proceedings of the 8th IEEE Real-Time Systems Symposium (RTSS '87), December 1-3, 1987, San Jose, Califorinia, USA*. IEEE Computer Society, 261–270.
[16] G. Lipari and E. Bini. 2003. Resource Partitioning among Real-Time Applications. In *15th Euromicro Conference on Real-Time Systems, 2003. Proceedings.* 151–158. https://doi.org/10.1109/EMRTS.2003.1212738
[17] C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM* 20, 1 (Jan. 1973), 46–61. https://doi.org/10.1145/321738.321743
[18] Jorge Martinez, Dakshina Dasari, Arne Hamann, Ignacio Sañudo, and Marko Bertogna. 2020. Exact Response Time Analysis of Fixed Priority Systems Based on Sporadic Servers. *Journal of Systems Architecture* 110 (Nov. 2020), 101836. https://doi.org/10.1016/j.sysarc.2020.101836
[19] C.W. Mercer, S. Savage, and H. Tokuda. 1993. Processor Capacity Reserves: An Abstraction for Managing Processor Usage. In *Proceedings of IEEE 4th Workshop on Workstation Operating Systems. WWOS-III.* 129–134. https://doi.org/10.1109/WWOS.1993.348160
[20] Alessandro Vittorio Papadopoulos, Martina Maggio, Alberto Leva, and Enrico Bini. 2015. Hard Real-Time Guarantees in Feedback-Based Resource Reservations. *Real-Time Systems* 51, 3 (June 2015), 221–246. https://doi.org/10.1007/s11241-015-9224-1
[21] S. Saewong, R.R. Rajkumar, J.P. Lehoczky, and M.H. Klein. 2002. Analysis of Hierarchical Fixed-Priority Scheduling. In *Proceedings 14th Euromicro Conference on Real-Time Systems. Euromicro RTS 2002.* 152–160. https://doi.org/10.1109/EMRTS.2002.1019197
[22] Lui Sha, R. Rajkumar, and S.S. Sathaye. 1994. Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems. *Proc. IEEE* 82, 1 (Jan. 1994), 68–82. https://doi.org/10.1109/5.259427
[23] Brinkley Sprunt, Lui Sha, and John Lehoczky. 1989. Aperiodic Task Scheduling for Hard-Real-Time Systems. *Real-Time Systems* 1, 1 (June 1989), 27–60. https://doi.org/10.1007/BF02341920
[24] Mark Stanovich, Theodore P. Baker, An-I Wang, and Michael Gonzalez Harbour. 2010. Defects of the POSIX Sporadic Server and How to Correct Them. In *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium.* 35–45. https://doi.org/10.1109/RTAS.2010.34
[25] Martin Stigge and Wang Yi. 2015. Graph-Based Models for Real-Time Workload: A Survey. *Real-Time Systems* 51, 5 (Sept. 2015), 602–636. https://doi.org/10.1007/s11241-015-9234-z
[26] J. K. Strosnider, J. P. Lehoczky, and Lui Sha. 1995. The Deferrable Server Algorithm for Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. *IEEE Trans. Comput.* 44, 1 (Jan. 1995), 73–91. https://doi.org/10.1109/12.368008
[27] Mohit Tawarmalani and Nikolaos V. Sahinidis. 2002. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming.* Nonconvex Optimization and Its Applications, Vol. 65. Springer US, Boston, MA. https://doi.org/10.1007/978-1-4757-3532-1