The Value of 'Even Distribution' for Temporal Resource Partitions

Yu Li, Albert M. K. Cheng

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number UH-CS-16-01

February 25, 2016

Keywords: Hierarchical Real-time Scheduling, Temporal Resource Partition, Real-time Task Scheduling, Resource Utilization, Schedulability Rate

Abstract

Most Hierarchical Real-time Scheduling (HiRTS) techniques have focused on temporal resource partitions which time units are periodically distributed. Although such periodic partitions could provide great flexibility for resource partitioning, engineers are stuck when trying to determine the schedulability of real-time tasks running on them. The main reason is that periodic partitions fail to effectively bound the difference between the ideal and the actual resource allocation. To solve this problem, some researchers introduced the Regular Partition, a type of temporal resource partition which is almost evenly distributed. Recent research has shown that it achieves maximal transparency for task scheduling. Some classical real-time scheduling problems on a regular partition can be easily transformed into equivalent problems on a dedicated single resource. However, the resource partitioning problem for regular partitions is much more complicated than the one for periodic partitions. Based on a practical 2-layer HiRTS platform, this paper first introduces new resource partitioning techniques for regular partitions. After that, it compares the overall performance of the periodic partition and the regular partition. We conclude that the regular partition is a better choice for the integration of real-time applications.

The Value of ‘Even Distribution’ for Temporal Resource Partitions

Yu Li, Albert M. K. Cheng

Abstract

Most Hierarchical Real-time Scheduling (HiRTS) techniques have focused on temporal resource partitions which time units are periodically distributed. Although such periodic partitions could provide great flexibility for resource partitioning, engineers are stuck when trying to determine the schedulability of real-time tasks running on them. The main reason is that periodic partitions fail to effectively bound the difference between the ideal and the actual resource allocation. To solve this problem, some researchers introduced the Regular Partition, a type of temporal resource partition which is almost evenly distributed. Recent research has shown that it achieves maximal transparency for task scheduling. Some classical real-time scheduling problems on a regular partition can be easily transformed into equivalent problems on a dedicated single resource. However, the resource partitioning problem for regular partitions is much more complicated than the one for periodic partitions. Based on a practical 2-layer HiRTS platform, this paper first introduces new resource partitioning techniques for regular partitions. After that, it compares the overall performance of the periodic partition and the regular partition. We conclude that the regular partition is a better choice for the integration of real-time applications.

Index Terms

Hierarchical Real-time Scheduling, Temporal Resource Partition, Real-time Task Scheduling, Resource Utilization, Schedulability Rate

I. INTRODUCTION

Aiming to integrate multiple real-time applications onto one single physical platform, hierarchical real-time scheduling (HiRTS) allows different real-time applications to share space or time on one computation resource. This problem is increasingly important as open systems [18] become more popular. Open systems make it easy to add and remove software applications as well as to increase resource utilization and reduce implementation cost when compared to systems which physically assign distinct computation resources to run different applications. In this paper, we focus on how to share time on one computation resource based on a practical 2-layer HiRTS system shown in Figure 1. A computation resource could be a single resource or an identical multiresource, and it is temporally divided into a group of Resource Partitions [2], which are managed by a global resource-level scheduler. On each resource partition, the real-time tasks belonging to a real-time application is scheduled by its own task-level scheduler.

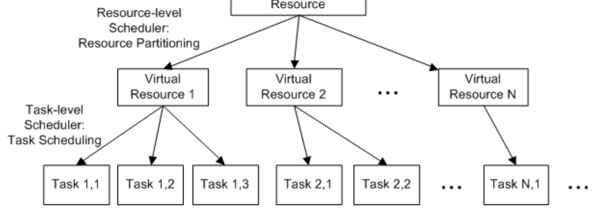


Fig. 1. A 2-Layer HiRTS System

The resource partition is an intermediate layer in this 2-layer system, and each resource partition only uses a fraction of the time on the computation resource. There are several HiRTS resource models. Typically, each HiRTS model has its own definition of resource partition containing some parameters. For example, a resource partition in the Periodic Model [10] exactly obtains c computation time units in each period p. A time unit is a system-defined unit of time for the purpose of scheduling, and there is no (either partition or task) preemption in it. Therefore, we may assume that all time parameters in this paper are integers without loss of generality.

The parameter values of a resource partition is called its signature. For example, periodic partition $(10, 4)$ represents a partition which obtains 4 time units in each period of 10 time units. The signature of a resource partition is the only sharing information between the global resource-level scheduler and its own task-level scheduler. The resource-level scheduler gathers all the partition signatures in the system and decides how to assign time units to the partitions. We call it the resource partitioning problem in HiRTS. Meanwhile, on each resource partition, real-time tasks are usually migrated from a non-hierarchical real-time system where they were directly running on a dedicated computation resource under a specific scheduling policy, such as Earliest Deadline First (EDF) or Rate Monotonic (RM) [12]. In most cases, the original schedulability tests for a dedicated resource do not work for a resource partition which only preempts a fraction of the computation time. Therefore, new task scheduling techniques have to be developed. We call it the task scheduling problem in HiRTS.

Then, we start to address the problems we shall discuss in this paper. At the resource level, a basic problem is how to schedule resource partitions on a single resource. We use (P1) Res-Single to represent this problem. In the multiresource scenario, there are two dominating categories of scheduling algorithms: global scheduling and partitioned scheduling. Global scheduling allows resource partitions to migrate between different computation resource units, while partitioned scheduling does not. They may dominate each other depending on different characteristics of computation resources. Generally, if the migration overhead is relatively small, global scheduling could provide higher efficiency; Otherwise, partitioned scheduling could perform better. Therefore, when considering the resource partitioning problem, we need to investigate both global scheduling and partitioned scheduling. We call them (P2) Res-Global and (P3) Res-Partitioned problems respectively.

For task scheduling, we shall discuss the schedulability problems for several popular task models. One task model we consider is the Periodic Task Model, in which two successive requests of each task are separated by exactly the same time interval, called its period. In this model, a task t_i is denoted by (c_i, p_i, d_i, o_i), where c_i, p_i, d_i and o_i are its execution time, period, deadline and offset, respectively. Most current HiRTS Models have investigated a simple case of the Periodic Task Model, in which each task's deadline is same as its period and offset is 0. We call (P4) Task-Periodic-Simple as the schedulability problem for this simple Periodic Task Model, and (P5) Task-Periodic-Generic as the one for the general Periodic Task Model. Another considered task model is the Sporadic Task Model, in which two successive requests of a task are separated by at least a time interval, called its minimum separation time. Similarly, we use (P6) Task-Sporadic to represent the schedulability problem for the Sporadic Task Model.

Related Work: There are several HiRTS resource models, such as the Regularity-based Model [1, 3], the Bounded-Delay Model [2, 3], the Periodic Model [10] and the EDP Model [11]. The Periodic Model (or Constant Bandwidth Server [16]) is the most popular one due to its simplicity for resource partitioning. Since a resource partition in this model is defined similarly to a periodic real-time task, the existing periodic task scheduling techniques can be used for periodic resource partitioning without changes. However, schedulability tests have been found only in the simplest case (P4) Task-Periodic-Simple for task scheduling in this model. Due to the blacked-out intervals (Figure 2) without any computation time available, researchers have been stuck in more complicated problems, such as (P5) Task-Periodic-Generic and (P6) Task-Sporadic.

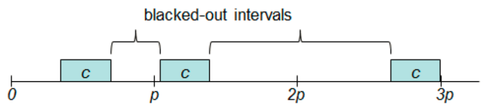


Fig. 2. Blacked-out Intervals on a Periodic Partition (p, c)

Besides the Periodic Model, other resource models had not received enough attention for a long time because they lacked effective resource partitioning algorithms. Recently, the Regularity-based Model has been greatly explored. Other than the Periodic Model, this model tends to evenly distribute the time units on a resource partition, and a parameter 'regularity' is used to restrain the time-unit distribution. A resource partition is called a Regular Partition when its regularity is minimal, which causes its time-unit distribution to be almost even. This idea was originally introduced in [7], and developed in [1, 3] for the single-resource scenario with some primitive results. Li and Cheng [4, 5] applied it onto a multiresource platform with global scheduling. Furthermore, they also comprehensively explored task scheduling on regular partitions in [15]. They found that some classical real-time scheduling problems on a regular partition can be easily transformed into equivalent problems on a dedicated single resource. Therefore, regular partitions are able to provide maximal transparency for task scheduling. In Table I, we summarize the current state of the art of the Regularity-based and the Periodic Models.

Problem	the Periodic Model	the Regularity-based Model
(P1) Res-Single	Converted to a Periodic Task Scheduling Problem	Primitive Results [1, 3]
(P2) Res-Global	Same As Above	Optimal Approximation Alogrithm (Magic7 [5])
(P3) Res-Partitioned	Same As Above	No Result
(P4) Task-Periodic-Simple	New Schedulability Tests for Periodic Partitions [10]	Converted to a Task Scheduling Problem on a Dedicated Resource [15]
(P5) Task-Periodic-Generic	No General Result	Same As Above
(P6) Task-Sporadic	No General Result	Same As Above

TABLE I
THE CURRENT STATE OF THE ART: THE PERIODIC MODEL AND THE REGULARITY-BASED MODEL

Contributions: One major contribution of this paper is to alleviate the weaknesses of the Regularity-based Model listed in Table I. The first weakness is that the current results on (P1) Res-Single are very primitive. Since the scheduling problem of regular partitions is announced NP-hard, the current solutions fall into the category of approximation algorithms. Mok and Feng [1, 3] gave an initial solution to schedule regular partitions on a single resource, where the weight of each regular partition is approximated by the values in an infinite sequence $\langle 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, ...\rangle$. However, they did not consider other approximation sequences, and their results have not been proved optimal. This paper is the first to comprehensively study (P1) Res-Single for the Regularity-based Model. It derives its schedulability bound for all feasible approximation sequences, and finds a group of sub-optimal approximation sequences which achieve higher resource utilization.

Based on the new results on (P1) Res-Single, this paper also studies (P3) Res-Partitioned in the Regularity-based Model for the first time, which is important because the migration overhead cannot be neglected in most real-time systems [13, 14]. It first derives the schedulability bound when a single approximation sequence is used. Then it introduces MulZ, a novel algorithm for (P3) Res-Partitioned, which drastically improves the overall resource utilization by using multiple approximation sequences. Even without considering the migration overhead due to global scheduling, MulZ outperforms the optimal global scheduling algorithm, especially on middle-to-large multiresource platforms. Moreover, we conclude that MulZ does not affect the original schedulability bound.

Another important contribution of this paper is to compare the overall performance of the Periodic and the Regularity-based Models. Since there has been no general result on (P5) Task-Periodic-Generic and (P6) Task-Sporadic in the Periodic Model, we can only compare the performance of (P4) Task-Periodic-Simple in both models with three different resource partitioning scenarios, (P1) Res-Single, (P2) Res-Global and (P3) Res-Partitioned, respectively. Our experimental result shows that the Regularity-based Model outperforms the Periodic Model in each scenario. Furthermore, considering the Regularity-based Model is able to handle comprehensive task models at the task level, it should be a better choice than the Partitioned Model when applying real-time application integrations, especially after we make up its weaknesses on resource partitioning in this paper.

Organization: The rest of this paper is organized as follows. We review the Regularity-based Model and its current approximation algorithm for (P2) Res-Global in Section II. We introduce our solutions for (P1)

Res-Single and (P3) Res-Partitioned in Sections III and IV, respectively. Section V presents our experimental results. Finally, we draw the conclusion in Section VI.

II. The Regular Partition and its Approximation Global Scheduling Algorithms

In this section, we review some basic definitions and important results in the Regularity-based Model introduced in [2, 5]. We make some changes or improvements to them for consistency and brevity. These prerequisite knowledge falls into two categories: one is about the properties of regular partitions; the other is about the current approximation algorithms based on global scheduling.

A. Regular Partition

The theoretical definition of a resource partition shows how time units are periodically assigned to it. As shown in Def. 2.1, it is specified by a period p and a time-unit sequence with length q. $W_P = \frac{q}{p}$ denotes the weight of P. In the Regularity-based Model, the weight of a resource partition is always a rational number between 0 and 1.

Definition 2.1 *A Resource Partition P is a tuple $(\mathcal{S}, p)$, where $\mathcal{S} = \langle s_1, s_2, ..., s_q : 0 \leq s_1 < s_2 < ... < s_q < p \rangle$ is the time-unit sequence; p is the period and p, q are co-prime.*

As shown in Figure 3, $S_P(t)$ denotes the Supply Function of P, equal to the total number of time units that are available in P from time 0 to t; $I_P(t) = S_P(t) - t \cdot W_P$ denotes the Instant Regularity of P, which shows the difference between the actual and ideal resource allocation on P at time t.

Lemma 2.1 *Suppose $P = (\mathcal{S}, p)$ is resource partition where q is the length of $\mathcal{S}$, then $\forall t \in [0, p)$,*

$$I_{\mathcal{A}}(t+1) - I_{\mathcal{A}}(t) = \begin{cases} 1 - \frac{q}{p} & if\ t \in \mathcal{S}; \\ -\frac{q}{p} & otherwise. \end{cases}$$

Proof: Immediately follows from the definition of Instant Regularity. ■

Definition 2.2 *A resource partition P is a Regular Partition if and only if $\forall t_1, t_2$, $|I_P(t_1) - I_P(t_2)| < 1$.*

A regular partition minimizes the deviation range of its instant regularity. As shown in Figure 3, the size of this range is limited to less than 1. We want to point out that this bounded range concept is very similar to the lag of a P-fair task [8, 9], but the lag range in P-fair is bounded by 2. Therefore, we cannot use the P-fair algorithm to schedule regular partitions because a regular partition has a much tighter restriction on the deviation range than a P-fair task.

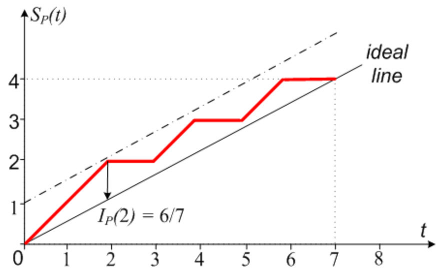


Fig. 3. A Regular Partition of Weight $\frac{4}{7}$

For a given regular partition P, let β_P denote the minimum value of P's instant regularities. Since $I_P(0) = 0$, by Def. 2.2, it is obvious that $\beta_P \in (-1, 0]$. Suppose p is P's period and $\frac{q}{p}$ is its weight where p, q are co-prime, then by Lemma 2.1, the possible values of β_P are $\{0, -\frac{1}{p}, -\frac{2}{p}, ..., -\frac{p-1}{p}\}$. Li and Cheng [15] prove that each value determines a $\frac{q}{p}$-weight regular partition. Therefore, there are exactly p different $\frac{q}{p}$-weight regular partitions.

Definition 2.3 *A regular partition P is a Standard Regular Partition when $\beta_P = 0$.*

Definition 2.4 *$T_0(p, q)$ represents the time-unit sequence of a standard regular partition with weight $\frac{q}{p}$.*

For example, Figure 3 shows $T_0(7, 4) = \langle 0, 1, 3, 5\rangle$. The motivation to point out such a special regular partition is that any other regular partition with the same weight can be easily obtained from it by applying a right-shift operation on its time-unit sequence. Therefore, we usually only need to check the properties of regular partitions on the standard ones. Figure 4 shows an example containing such kind of right shifts.

Definition 2.5 *$T(p, q, \delta)$ represents the time-unit sequence right shifted from $T_0(p, q)$ by δ time units, where a modulus operation is applied when a time unit is out of $[0, p)$. Specially, $T(p, q, 0) = T_0(p, q)$.*

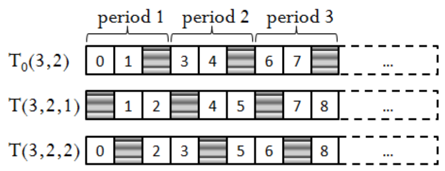


Fig. 4. Right Shifting Regular Partition $(T_0(3, 2), 3)$

The following discussion answers the question of whether a regular partition can be a part of another one in some special cases. These results are very important for our approximation method of regular partition scheduling. We always use it to determine counter examples when we check the feasibility of an approximation sequence. In this paper, we formally define a partial order $\prec_{rp}$ to indicate the containment relation between regular partitions.

Definition 2.6 *$\frac{q_1}{p_1} \prec_{rp} \frac{q_2}{p_2}$ if and only if $\exists\delta$, $T(p, \frac{q_1}{p_1} \cdot p, \delta) \subset T_0(p, \frac{q_2}{p_2} \cdot p)$, where $p = LCM(p_1, p_2)$.*

For convenience, in this paper, we use RP_w to represent a regular partition whose weight is w. Then, we can also describe $\prec_{rp}$ like this: $w_1 \prec_{rp} w_2$ if and only if RP_{w_1} could be contained by RP_{w_2}. This containment relation is checked frequently while scheduling regular partitions. The following two lemmas investigate this containment relation in some special cases.

Lemma 2.2 *If $w_1 \prec_{rp} w_2$ where $w_1 + w_2 < 1$ and $w_2 < 2w_1$, then $\exists n > 0$, $w_1 = \frac{n+1}{4n+3}$, $w_2 = \frac{2n+1}{4n+3}$.*
Proof: Lemma 4.3 in [5] shows the same property. The proof is also presented Appendix B. ■

Lemma 2.3 *If $w_2 \leq \frac{1}{3}$ and $\frac{w_2}{2} < w_1 < w_2$, $w_1 \not\prec_{rp} w_2$.*
Proof: Immediately follows from Lemma 2.2. ■

For example, by Lemma 2.3, we easily conclude that a $\frac{1}{3}$-weight regular partition does not contain any $\frac{1}{4}$-weight one; by Lemma 2.2, a $\frac{5}{9}$-weight regular partition does not contain any $\frac{1}{3}$-weight one.

B. Approximation Algorithms for Regular Partitions' Global Scheduling

The problem is: given $\{w_i : 1 \leq i \leq n\}$ as the weights of n regular partitions, how to schedule them on an identical multiresource with global scheduling? An approximation method [5] adjusts each w_i to the closest greater or equal value in an Approximation Boundary Sequence (ABS) or an Extended Approximation Boundary Sequence (E-ABS) that are defined as follows:

Definition 2.7 *An Approximating Boundary Sequence (ABS) is an infinite number sequence $\langle b_1, b_2, b_3, ...\rangle$, where $\forall i,\ 0 < b_{i+1} < b_i < 1$; $\lim b_n = 0$ when $n \to \infty$.*

Definition 2.8 *An Extended Approximating Boundary Sequence (E-ABS) is a tuple (B, B'), where ABSes $B = \langle b_1, b_2, ...\rangle$, $B' = \langle b'_1, b'_2, ...\rangle$, and $b_1 + b'_1 < 1$.*

E-ABS is introduced to achieve lower approximation overhead than ABS because it could contain more and denser elements used for approximation. For a given E-ABS $E = (B, B')$, we say $b \in E$ if and only if $b \in B$ or $1 - b \in B'$. Sometimes we use $\langle ..., 1 - b'_2, 1 - b'_1, b_1, b_2, ...\rangle$ to represent an E-ABS for convenience, though it is not a formal representation for sequences. The following are some important functions for a given ABS/E-ABS B originally defined in [5]. Specially, if B is an ABS, let B include 1 for boundary control.

Definition 2.9 Approximation Function $R_B(w)$:

$R_B(w) = \min\{b : b \in B; b \geq w\}$.

$R_B(w)$ approximates a weight w at the closest greater or equal value in B.

Definition 2.10 Schedulability Bound Υ_B:

$\Upsilon_B = \min\{\frac{b}{b'} : b, b' \in B; b < b'; \nexists b'' \in B, b < b'' < b'\}$.

Υ_B equals the minimal quotient of any two consecutive elements in B, which indicates the low-bound resource utilization due to the approximation strategy by B.

Definition 2.11 Approximation Overhead $\mathcal{O}_B$:
$\mathcal{O}_B$ determines the average resource utilization of B. We do not present the equation of $\mathcal{O}_B$ here because this paper does not use this equation. In Figure 5, the stairs shape indicates $R_B(w)$, and $\mathcal{O}_B$ equals the area of the slash-filled triangles.

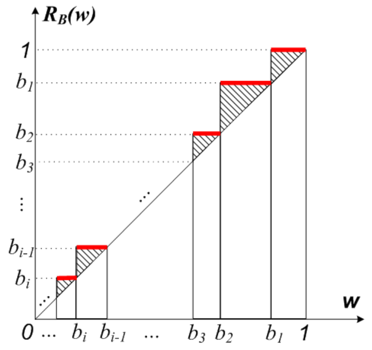


Fig. 5. Approximation Function and Overhead of an ABS

Let us make a naming convention. In this paper, we use m-resource to represent a multiresource with m identical resource-units. Specially, *1-resource* represents a single-resource. Then, we can define the feasibility of an ABS/E-ABS in Def. 2.12, where $\{n_i \times w_i : i = 1, 2, ..., k\}_{rp}$ denotes a partition group containing n_i times RP_{w_i}, and $1 \times w_i$ can be simplified to w_i.

Definition 2.12 *An ABS/E-ABS B is globally-feasible if and only if $\forall b_i \in B$ $(i = 1, 2, ..., n)$ where $\sum_{i=1}^{n} b_i \leq m$, $\{b_i : i = 1, 2, ..., n\}_{rp}$ is schedulable on an m-resource via global scheduling.*

The thought behind "B is globally-feasible" is: For a group of regular partitions $\{w_i : i = 1, 2, ..., n\}_{rp}$, we first approximate the weight of each partition at its closest boundary in B using $R_B(w)$. If the sum of these approximated weights does not exceed m, they are always schedulable on an m-resource via global scheduling. On the contrary, if we are able to find a partition group as a counter example that is unschedulable after being approximated by B via global scheduling, we can claim that B is not globally-feasible.

Name	*Definition*	*Globally-Feasible*
$\mathcal{G}_{n,m}$	$\langle \frac{1}{n\cdot m}, \frac{1}{n\cdot m^2}, \frac{1}{n\cdot m^3}, ...\rangle$	yes
$\mathcal{H}_{n,m}$	$\langle \frac{n-1}{n}, \frac{n-2}{n}, ..., \frac{1}{n}, \frac{1}{n\cdot m}, \frac{1}{n\cdot m^2}, \frac{1}{n\cdot m^3}, ...\rangle$	iff $n \in$ RMN *
$\mathcal{Z}_{n,m}$	$(\mathcal{H}_{n,m}, \mathcal{G}_{n,m})$	iff $n \in$ RMN

TABLE II
TYPICAL ABSES AND E-ABSES

Table II describes some types of ABSes/E-ABSes defined in [5] and their feasibility for global scheduling, and we also list some specific ABSes/E-ABSes as follows, which are widely used in our later discussion. Specially, $\mathcal{Z}_{7,2}$ is the optimal ABS/E-ABS for global scheduling found in [5].

* RMN (Regularity Magic Numbers) is an integer set {2,3,4,5,7}.

$\mathcal{G}_{1,2} = \langle \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, ... \rangle$
$\mathcal{Z}_{2,2} = \mathcal{Z}_{4,2} = \langle ..., \frac{31}{32}, \frac{15}{16}, \frac{7}{8}, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, ... \rangle$
$\mathcal{Z}_{3,2} = \langle ..., \frac{23}{24}, \frac{11}{12}, \frac{5}{6}, \frac{2}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$
$\mathcal{Z}_{5,2} = \langle ..., \frac{39}{40}, \frac{19}{20}, \frac{9}{10}, \frac{4}{5}, \frac{3}{5}, \frac{2}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{20}, \frac{1}{40}, ... \rangle$
$\mathcal{Z}_{7,2} = \langle ..., \frac{55}{56}, \frac{27}{28}, \frac{13}{14}, \frac{6}{7}, \frac{5}{7}, \frac{4}{7}, \frac{3}{7}, \frac{2}{7}, \frac{1}{7}, \frac{1}{14}, \frac{1}{28}, \frac{1}{56}, ... \rangle$

At the end of this review section, we list the symbols used in this paper in Table III.

B, B', A	ABS or E-ABS
b, b', b'', b_i	item in ABS or E-ABS
w, w_i	weight of regular partition
$R_B(w)$	approximation function of B
Υ_B	schedulability bound of B
$\mathcal{O}_B$	approximation overhead of B
$T_0(p, q)$	time-unit sequence of standard regular partition
$T(p, q, \delta)$	time-unit sequence of regular partition
RP_w	regular partition of weight w
$\prec_{rp}$	containment relation between regular partitions
G, $\{...\}_{rp}$	regular partition group

TABLE III
SYMBOL TABLE

III. SINGLE-RESOURCE SCHEDULING FOR REGULAR PARTITIONS

Although there already have been some approximation algorithms for single-resource scheduling of regular partitions, such as AAF [2] and Magic7-Single [5], these algorithms have not been proved *optimal*. We will deeply study this problem by also probing into approximation algorithms in this section. First, we define the ABS/E-ABS feasibility on a single resource as follows. For convenience, we say a regular partition group is *on-1-schedulable* if it is schedulable on a single resource.

Definition 3.1 *An ABS/E-ABS B is on-1-feasible if and only if $\forall b_i \in B$ $(i = 1, 2, ..., n)$ where $\sum_{i=1}^{n} b_i \leq 1$, $\{b_i : i = 1, 2, ..., n\}_{rp}$ is on-1-schedulable.*

This definition is very similar to Def. 2.12 for *globally-feasible*, but *on-1-feasible* only requires that an ABS/E-ABS always works on a 1-resource. Therefore, "*B is globally-feasible*" is sufficient but not necessary for "*B is on-1-feasible*". There are some ABSes/E-ABSes that are *on-1-feasible* but not *globally-feasible*. $\langle \frac{2}{3}, \frac{1}{2}, \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$ is such an example. We formally examine it as follows.

Observation 1 $\langle \frac{2}{3}, \frac{1}{2}, \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$ *is on-1-feasible.*

Proof: Let B denote the given ABS. Following Def. 3.1, we only need to prove that $\forall b_i \in B$ $(b_1 \geq b_2 \geq ... \geq b_n)$ where $\sum_{i=1}^{n} b_i \leq 1$, $G = \{b_i : i = 1, 2, ..., k\}_{rp}$ is on-1-schedulable. CASE 1: $b_1 \leq \frac{1}{6}$. In Table II, we know $\mathcal{G}_{3,2} = \langle \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$ is globally-feasible, then it is also on-1-feasible. It follows that G is on-1-schedulable in this case. CASE 2: $b_1 = \frac{1}{2}$. If $b_2 = \frac{1}{2}$, it is easy to schedule G; otherwise, we need to examine whether $G' = \{b_i : i = 2, ..., k\}_{rp}$ is schedulable on a $\frac{1}{2}$-weight regular partition, which is equivalent to that $G'' = \{2b_i : i = 2, ..., k\}_{rp}$ is on-1-schedulable, where all items in G'' belong to $\langle \frac{1}{3}, \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$. From Table II, $\mathcal{H}_{3,2} = \langle \frac{2}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$ is on-1-feasible, and it follows that G'' is on-1-schedulable. CASE 3: $b_1 = \frac{2}{3}$. It is obvious that G does not contain $\frac{1}{2}$. Since $\mathcal{H}_{3,2} = \langle \frac{2}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$ is on-1-feasible, G is on-1-schedulable. ■

Observation 2 $\langle \frac{2}{3}, \frac{1}{2}, \frac{1}{6}, \frac{1}{12}, \frac{1}{24}, ... \rangle$ *is not globally-feasible.*

Proof: We only need to show that partition group $G = \{2 \times \frac{2}{3}, \frac{1}{2}\}_{rp}$ is unschedulable on a 2-resource. Figure 4 presents the three forms that $\frac{2}{3}$-weight regular partitions have. CASE 1: If the two $\frac{2}{3}$-weight regular partitions in G is same, without loss of generality, suppose their time-unit sequences are both $T_0(3, 2)$. Obviously, the remaining time units (a couple of $\langle 2, 5, 8, ...\rangle$) cannot produce a $\frac{1}{2}$-weight regular partition, whose time-unit sequence should be $\langle 0, 2, 4, ...\rangle$ or $\langle 1, 3, 5, ...\rangle$. CASE 2: Otherwise, without loss of generality, suppose G contains the first two forms of partitions in Figure 4, and the remaining time units $\langle 0, 2, 3, 5, 6, 8, ...\rangle$ also cannot produce a $\frac{1}{2}$-weight regular partition. ■

A. Schedulability Bound on a Single Resource

We figure out the upper limit of the schedulability bound of an on-1-feasible approximation sequence in Theorem 3.1 by proving that it is not on-1-feasible if its schedulability bound exceeds 0.5.

Lemma 3.1 *$\forall$ on-1-feasible ABS/E-ABS B where $\Upsilon_B > 0.5$, $\exists p \geq 3$, $\frac{1}{p} \in B$.*

Proof: presented in Appendix C. ■

Lemma 3.2 *$\forall p > 1$, after scheduling $(p-1)$ times $\frac{1}{p}$-weight regular partitions on a 1-resource, the remaining time units compose another $\frac{1}{p}$-weight regular partition.*

Proof: It is true because $T(p, 1, \delta) = \langle \delta \rangle$ for $\delta \in [0, p)$. ■

Theorem 3.1 *$\forall$ on-1-feasible ABS/E-ABS B, $\Upsilon_B \leq 0.5$.*

Proof: If $\Upsilon_B > 0.5$, by Lemma 3.1, $\exists p \geq 3$, $\frac{1}{p} \in B$. And by Def. 2.10, $\exists w \in B, w \in (\frac{1}{2p}, \frac{1}{p})$. Then $\{(p-1) \times \frac{1}{p}, w\}_{rp}$ is not on-1-schedulable because $w \not\prec_{rp} \frac{1}{p}$ (Lemmas 3.2, 2.3). This is a counter example against that B is on-1-feasible. It follows that Υ_B cannot be great than 0.5. ■

B. Sub-optimal Approximation on Single Resources

Next, we start to consider the approximation overhead. Based on the shapes shown in Figure 5, intuitively, the approximation overhead will decrease if we add more elements into the sequence. So we first define the containment relation between approximation sequences.

Definition 3.2 *Suppose A,B are ABSes/E-ABSes, $A \subseteq B$ if $\forall b \in A$, $b \in B$; $A \subset B$ if $A \subseteq B$ but $B \nsubseteq A$.*

The next conclusion is obvious from Figure 5:

Lemma 3.3 *Suppose A,B are ABSes/E-ABSes, $\mathcal{O}_A < \mathcal{O}_B$ if $A \subset B$.*

Lemma 3.3 shows that a feasible approximation sequence can reach its minimum approximation overhead if we cannot add any element into it without violating its feasibility. This conclusion leads to the idea of 'Saturated'.

Definition 3.3 *An ABS/E-ABS A is <u>saturated</u> if A is on-1-feasible, and $\forall b \notin A$, $A \cup \langle b \rangle$ is not on-1-feasible.*

Furthermore, we say a saturated approximation sequence is sub-optimal if it also reaches the maximal schedulability bound.

Definition 3.4 *An ABS/E-ABS A is <u>sub-optimal</u> if A is saturated and $\Upsilon_A = 0.5$.*

In this paper, we find a group of sub-optimal E-ABSes based on the idea of Regularity Magic Numbers [5]. Reminder that $\mathcal{Z}_{n,2}$ has been defined in Table II.

Theorem 3.2 *$\mathcal{Z}_{n,2}$ is sub-optimal if $n \in \{3, 4, 5, 7\}$.*

Proof: It is obvious that $\Upsilon_{\mathcal{Z}_{n,2}} = 0.5$ and $\mathcal{Z}_{n,2}$ is on-1-feasible. We only need to prove that $\mathcal{Z}_{n,2}$ is saturated. Since $\mathcal{Z}_{n,2} = \langle ..., \frac{4n-1}{4n}, \frac{2n-1}{2n}, \frac{n-1}{n}, \frac{n-2}{n}, ..., \frac{1}{n}, \frac{1}{2n}, \frac{1}{4n}, ...\rangle$, $\forall b \notin \mathcal{Z}_{n,2}$, there are four cases:
CASE 1: $b \in (\frac{1}{2^k n}, \frac{1}{2^{k-1} n})$ where $k > 0$.
$\{(2^{k-1}n - 1) \times \frac{1}{2^{k-1}n}, b\}_{rp}$ is not on-1-schedulable because $b \not\prec_{rp} \frac{1}{2^{k-1}n}$ (Lemmas 3.2, 2.3).
CASE 2: $b \in (1 - \frac{1}{2^{k-1}n}, 1 - \frac{1}{2^k n})$ where $k > 0$.
$\{b, \frac{1}{2^k n}\}_{rp}$ is not on-1-schedulable because $\frac{1}{2^k n} \not\prec_{rp} (1 - b)$ (Lemmas 3.2, 2.3).

CASE 3: $b \in (\frac{k-1}{n}, \frac{k}{n})$ and $b + \frac{k}{n} < 1$, where $1 < k < n$.
$\{1 - \frac{k}{n}, b\}_{rp}$ is on-1-schedulable $\Rightarrow b \prec_{rp} \frac{k}{n}$
$\Rightarrow \exists m > 0,\ b = \frac{m+1}{4m+3}$ and $\frac{k}{n} = \frac{2m+1}{4m+3}$ (Lemma 2.2)
$\Rightarrow m = 1, n = 7, k = 3, b = \frac{2}{7}$.
This result contradicts $b > \frac{k-1}{n}$.
CASE 4: $b \in (\frac{k-1}{n}, \frac{k}{n})$ and $b + \frac{k}{n} > 1$, where $1 < k < n$.
$\{b, 1 - \frac{k}{n}\}_{rp}$ is on-1-schedulable $\Rightarrow (1 - \frac{k}{n}) \prec_{rp} (1 - b)$
$\Rightarrow \exists m > 0,\ \frac{k}{n} = \frac{3m+2}{4m+3}$ and $b = \frac{2m+2}{4m+3}$ (Lemma 2.2)
$\Rightarrow m = 1, n = 7, k = 5, b = \frac{4}{7}$.
This result also contradicts $b > \frac{k-1}{n}$. ■

IV. Partitioned Multiresource Scheduling for Regular Partitions

We have deeply studied the single-resource scheduling problem for regular partitions. A maximum schedulability bound and some sub-optimal E-ABSes have been found. Next, we start to investigate the partitioned multiresource scheduling problem for regular partitions.

There are two steps in a common partitioned scheduling algorithm: (1) allocate resource partitions (or real-time tasks) to resource-units; and (2) schedule them on each resource-unit. Step 1 has two concerns when allocating a resource partition: one is which resource-units can contain it; the other is that if multiple resource-units can do it, which one should be chosen. These two issues correspond to the single-resource scheduling algorithm and the allocation algorithm, respectively.

We adopt the naming convention in [6], where *SA-RA* denotes the partitioned scheduling algorithm combining a reasonable allocation algorithm *RA* and a single-resource scheduling algorithm *SA*. For example, $\mathcal{Z}_{3,2}$-*WF* represents the combination of the Worst Fit First resource allocation algorithm and the approximation single-resource scheduling algorithm using $\mathcal{Z}_{3,2}$. Also, $\Upsilon_{\textit{SA-RA}}$ denotes the Schedulability Bound of *SA-RA*.

A. Using a Single Approximation Sequence

There are some particularity when scheduling regular partitions. Since we use an approximation methodology for single-resource scheduling, we can approximate the regular partitions before the allocation step. If the approximation algorithm is based-on a single approximation sequence, we only need to guarantee that the total weight allocated to each resource-unit does not exceed 1. We easily conclude that the schedulability bound of such a partitioned algorithm cannot exceed 0.5.

Theorem 4.1 $\forall$ *ABS/E-ABS* B, $\Upsilon_{B\textit{-RA}} \leq \Upsilon_B$.

Proof: Since the weight of each regular partition has to be approximated by B, no matter how these partitions are allocated, $\Upsilon_{B\textit{-RA}}$ cannot exceed Υ_B. ■

Corollary 4.1 $\forall$ *ABS/E-ABS* B,$\Upsilon_{B\textit{-RA}} \leq 0.5$.

We assume that the weight of each regular partition is static, such that we can sort the regular partitions by their weights before the allocation step. Some allocation algorithms are based on this assumption, such as First Fit Decreasing (*FFD*) and Best Fit Increasing (*BFI*). We first notice that $\mathcal{G}_{1,2}$-*FFD* and $\mathcal{Z}_{2,2}$-*FFD* reach the maximal schedulability bound shown in Corollary 4.1.

Lemma 4.1 $\Upsilon_{\mathcal{G}_{1,2}\textit{-FFD}} = 0.5$.

Proof: From Table II, $\mathcal{G}_{1,2} = \langle \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, ... \rangle$, and $\Upsilon_{\mathcal{G}_{1,2}} = 0.5$. Suppose $\{n_i \times \frac{1}{2^i} : i = 1, 2, 3, ...\}_{rp}$ is the approximated partition set. Obviously, these partitions can be successfully allocated to an m-resource by *FFD* when $\sum_{i>0} \frac{n_i}{2^i} \leq m$. Therefore, $\mathcal{G}_{1,2}$-*FFD* always works when the total weight of the original partitions is not greater than 0.5. It follows $\Upsilon_{\mathcal{G}_{1,2}\textit{-FFD}} \geq 0.5$. By Corollary 4.1, $\Upsilon_{\mathcal{G}_{1,2}\textit{-FFD}} = 0.5$. ■

Lemma 4.2 $\Upsilon_{\mathcal{Z}_{4,2}\textit{-FFD}} = 0.5$.

Proof: $\mathcal{Z}_{4,2} = \langle ..., \frac{7}{8}, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, ... \rangle$. Let's compare the scheduling of $\mathcal{Z}_{4,2}$-*FFD* and $\mathcal{G}_{1,2}$-*FFD*. Suppose $\{N_i \times (1 - \frac{1}{2^i}) : i = 2, 3, ...; n_i \times \frac{1}{2^i} : i = 1, 2, 3, ...\}_{rp}$ is the approximated partition set by $\mathcal{Z}_{4,2}$, then $\{(\sum_{i>1} N_i) \times 1;\ n_i \times \frac{1}{2^i} : i = 1, 2, 3, ...\}_{rp}$ is the one by $\mathcal{G}_{1,2}$. Figure 6 shows the partition allocations of these two algorithms.

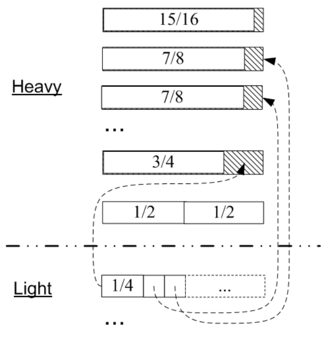


Fig. 6. $\mathcal{Z}_{4,2}$-*FFD* and $\mathcal{G}_{1,2}$-*FFD*

Their allocations of heavy partitions (weight $\geq \frac{1}{2}$) are exactly the same. The difference is between the light-partition allocations. Since $\mathcal{Z}_{4,2}$-*FFD* leaves some blanks in the heavy part, some light partitions could be assigned into these blanks. Nevertheless, these changes do not negatively impact the schedulability. ■

Then we prove that each approximation sequence with the maximal schedulability bound has to contain $\frac{1}{2}$.

Lemma 4.3 *If $\frac{1}{2} \notin B$, then $\forall$ RA, $\Upsilon_{B\text{-}RA} \leq \max\{0, b : b \in B, b < \frac{1}{2}\}$.*

Proof: Let $b_1 = \max\{0, b : b \in B, b < \frac{1}{2}\}$ and $b_2 = \min\{1, b : b \in B, b > \frac{1}{2}\}$. Schedule $\{(m+1) \times (b_1 + \epsilon)\}_{rp}$ on an m-resource with B, where $\epsilon < b_2 - b_1$. This set is approximated to $\{(m+1) \times b_2\}_{rp}$. It is unschedulable for any allocation algorithm because $b_2 > \frac{1}{2}$. It follows $\Upsilon_{B\text{-}RA} \leq \frac{(m+1)\cdot(b_1+\epsilon)}{m} \to b_1$ when $m \to \infty$ and $\epsilon \to 0$. Therefore, $\Upsilon_{B\text{-}RA} \leq b_1$. ■

Corollary 4.2 *If $\frac{1}{2} \notin B$, then $\forall RA$, $\Upsilon_{B\text{-}RA} < 0.5$.*

Theorem 4.2 shows that $\mathcal{Z}_{4,2}$ is the only sub-optimal sequence reaching the maximal schedulability bound.

Lemma 4.4 *Given an on-1-feasible ABS/E-ABS B where $\Upsilon_B = 0.5$, if $\frac{1}{2} \in B$, then $\forall k > 0, \frac{1}{2^k} \in B$ and $\forall w \in (\frac{1}{2^{k+1}}, \frac{1}{2^k})$, $w \notin B$.*

Proof: Use an inductive method. When $k = 1$, $\frac{1}{2} \in B$. Assume $\exists w \in (\frac{1}{4}, \frac{1}{2})$, $w \in B$. Check the schedulability of $\{\frac{1}{2}, w\}_{rp}$ on a 1-resource. Suppose $RP_{\frac{1}{2}}$ preempts all the time units at even numbers. No matter in which case among $\frac{1}{w} \in (2, 3)$, $\frac{1}{w} \in (3, 4)$ and $\frac{1}{w} = 3$, there exist two consecutive time units on RP_w whose distance is 3. It follows that $\{\frac{1}{2}, w\}_{rp}$ is not on-1-schedulable because all the time units at even numbers are already preempted by $RP_{\frac{1}{2}}$. Thus, $\forall w \in (\frac{1}{4}, \frac{1}{2})$, $w \notin B$. When $k > 1$, by the inductive assumption, $\frac{1}{2^{k-1}} \in B$ and $\forall w \in (\frac{1}{2^k}, \frac{1}{2^{k-1}})$, $w \notin B$. Then $\frac{1}{2^k} \in B$ (otherwise, $\Upsilon_B < 0.5$ by Def. 2.10). Assume $\exists w \in (\frac{1}{2^{k+1}}, \frac{1}{2^k})$, $w \in B$. Let $p = 2^k \geq 4$, then $\{(p-1) \times \frac{1}{p}, w\}_{rp}$ is not on-1-schedulable because $w \not\prec_{rp} \frac{1}{p}$ (Lemmas 3.2, 2.3). Therefore, $\forall w \in (\frac{1}{2^{k+1}}, \frac{1}{2^k})$, $w \notin B$. ■

Theorem 4.2 *If B is a sub-optimal ABS/E-ABS and $\Upsilon_{B\text{-}FFD} = 0.5$, then $B = \mathcal{Z}_{4,2}$.*

Proof: By Lemma 4.2 and Corollary 4.2, we only need to show that $\mathcal{Z}_{4,2}$ is the only sub-optimal ABS/E-ABS containing $\frac{1}{2}$. Suppose B is such a sequence. By Lemma 4.4, $\forall k > 0, \frac{1}{2^k} \in B$ and $\forall w \in (\frac{1}{2^{k+1}}, \frac{1}{2^k})$, $w \notin B$. On the other hand, $\forall k > 0$, $\forall w \in (1 - \frac{1}{2^k}, 1 - \frac{1}{2^{k+1}})$, $w \notin B$; otherwise, $\{w, \frac{1}{2^{k+1}}\}_{rp}$ is on-1-schedulable $\Rightarrow \frac{1}{2^{k+1}} \prec_{rp} (1 - w)$, which contradicts Lemma 2.2. It follows $B \subseteq \mathcal{Z}_{4,2}$. Since B is saturated, $B = \mathcal{Z}_{4,2}$. ■

B. Using Multiple Approximation Sequences

The performance of a partitioned scheduling algorithm strongly depends on its approximation overhead. For a given weight, the approximation overhead is not the same on different approximation sequences. For example, $R_{\mathcal{Z}_{4,2}}(0.45) = 0.5$ and $R_{\mathcal{Z}_{7,2}}(0.45) = 0.57$. This fact inspires us to use multiple approximation sequences in a partitioned scheduling algorithm. We call it MulZ when we use $\mathcal{Z}_{3,2}$, $\mathcal{Z}_{4,2}$, $\mathcal{Z}_{5,2}$, $\mathcal{Z}_{7,2}$ simultaneously, and present its pseudocode as follows.

```
(0)   resource-units $R := \{R_j\{factor = 0, rest = 1\} : j \in [1, m]\}$;
```

```
(1)  partitions $P := \{P_j\{weight, res\text{-}unit = 0\} : j \in [1, s]\}$;
----------------------------------------
(2)  bool MulZ_FFD()
(3)    sort $P$ in non-increasing order;
(4)    for $j = 1$ to $s$ do
(5)      $P_j$.resource := MulZ_FFD_Alloc($P_j$.weight);
(6)      if $P_j$.resource $= 0$ return false;
(7)    od
(8)    return true;
----------------------------------------
(9)  int MulZ_FFD_Alloc($w$)
(10)   for $i = 0; n \in \{3, 4, 5, 7\}; i$++ do
(11)       $A_i = R_{\mathcal{Z}_{n,2}}(w)$;
(12)   od
(13)   for $k = 1$ to $4$ do
(14)       $r$ := the $k$-th minimum item in array $A$;
(15)       $f := n$, where $w$ is approximated at $r$ by $\mathcal{Z}_{n,2}$;
(16)       for $j = 1$ to $m$ do
(17)           if $R_j$.factor = $f$ and $R_j$.rest $\geq r$ do
(18)               $R_j$.rest := $R_j$.rest $- r$;
(19)               return j;
(20)           od
(21)           else if $R_j$.factor = 0 do
(22)               $R_j$.factor := $f$;
(23)               $R_j$.rest := $1 - r$;
(24)               return j;
(25)           od
(26)       od
(27)   od
(28)   return 0;
```

The first two lines define and initialize the data structures. In line (0), a positive value of "*factor*", n, indicates that a resource-unit is not empty and the partitions on it are approximated by $\mathcal{Z}_{n,2}$. In line (1), a positive value of "*res-unit*", m, indicates that a partition is assigned to the m-th resource-unit. Function *MulZ_FFD* first sorts the resource partitions in non-increasing order, and then calls *MulZ_FFD_Alloc* within a loop to allocate resource for each partition. Lines 10-12 compute the approximated weights of the sub-optimal E-ABSes, and store them as an array. The loop of lines 16–26 checks the availability of each resource-unit one by one for the current partition, where line 14 determines which E-ABS is chosen for approximation in the current iteration; lines 17–20 search available non-empty resource-units using the chosen E-ABS; lines 21–25 assign the current partition to an empty resource-unit if the condition in line 17 fails for all non-empty resource-units (always having lower indexes in R). If there are empty resource-units remaining when starting function *MulZ_FFD_Alloc*, the loop of lines 16–26 must terminate at either line 19 or line 24 when $k = 1$. Therefore, the branch of lines 21–25 is only executed when $k = 1$. It follows that an empty resource-unit always chooses its working E-ABS such that the minimum approximation overhead is achieved for its first assigned partition.

***A Partitioning Example of MulZ-FFD*:**
Partition $G = \{0.65, 0.6, 0.55, 0.5, 0.35, 0.3, 3 \times 0.25\}_{rp}$ on a 4-resource. Let $U(G, n) = \sum_{w \in G} R_{\mathcal{Z}_{n,2}}(w)$. It is easy to check that $\forall n \in \{3, 4, 5, 7\}$, $U(G, n) > 4$. Therefore, any single $\mathcal{Z}_{n,2}$ does not work in this scenario, even if we use a global scheduling strategy without considering migration overhead. However, MulZ-FFD is able to do that. Let's see how the first partition 0.65 is assigned. When $n = 3, 4, 5, 7$, its approximated weight equals $\frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{7}$, respectively. We choose the minimum one $\frac{2}{3}$, and the working E-ABS is $\mathcal{Z}_{3,2}$. Then we assign this partition to the No. 0 resource-unit because currently all the resource-units are empty. Meanwhile, the working E-ABS on the No. 0 resource-unit is set to $\mathcal{Z}_{3,2}$. The rest can be done in the same manner, and the final result is shown in Table IV. The overall resource utilization is 92.5%. Notice that although $\mathcal{Z}_{7,2}$ cannot achieve the minimum approximated overhead for the last remaining 0.25-weight partition, this partition is still assigned to the No. 2 resource-unit because the other three resource-units cannot accommodate it at that moment.

MulZ-FFD is an intuitive algorithm, which is unable to promise optimal performance theoretically. Nevertheless, our experimental results in the next section show that it has better performance than the current optimal global scheduling algorithm. Meanwhile, Theorem 4.3 shows that the maximal schedulabiltiy bound 0.5 is still kept.

Lemma 4.5 *Suppose weights* $w_1, w_2, ..., w_s$ $(w_1 \geq w_2 \geq ... \geq w_s > 0)$ *are already assigned to a resource-unit by MulZ-FFD, where* $\mathcal{Z}_{n,2}$ $(n \in \{3,4,5,7\})$ *is its working E-ABS, and* $\sum_{i=1}^{s} w_i < 0.5$. $\forall w_{s+1} \in (0, w_s]$*, this resource-unit is still able to accommodate* w_{s+1}.

Proof: We only check $\mathcal{Z}_{7,2}$ here. The others can be checked similarly. Let $r_i = R_{\mathcal{Z}_{7,2}}(w_i)$ be the approximated weight of w_i for $i = 1, 2, ..., s+1$. Notice $w_i \leq r_i < 2w_i$. Since $\mathcal{Z}_{7,2}$ is sub-optimal, we need to show $U = \sum_{i=1}^{s+1} r_i \leq 1$.
CASE 1: $w_1 \in (\frac{3}{7}, \frac{1}{2})$, $r_1 = \frac{4}{7}$. This case is impossible. When *MulZ-FFD* assigns w_1 to an empty resource-unit, it should choose $\mathcal{Z}_{4,2}$ as the working E-ABS because $R_{\mathcal{Z}_{4,2}}(w_1) = \frac{1}{2} < r_1$.
CASE 2: $w_1 \in (\frac{2}{7}, \frac{3}{7}]$, $r_1 = \frac{3}{7}$. Similarly, to choose $\mathcal{Z}_{7,2}$ as the working E-ABS when assigning w_1, w_1 must be in $(\frac{2}{5}, \frac{3}{7}]$. If $s = 1$, then $U \leq 2r_1 < 1$; otherwise, $s > 1 \Rightarrow \sum_{i=2}^{s} w_i < 0.5 - w_1 < 0.1 \Rightarrow U < r_1 + 2\sum_{i=2}^{s} w_i + 2w_{s+1} \leq \frac{3}{7} + 4\sum_{i=2}^{s} w_i < 1$.
CASE 3: $w_1 \in (\frac{1}{7}, \frac{2}{7}]$, $r_1 = \frac{2}{7}$. To choose $\mathcal{Z}_{7,2}$, w_1 must be in $(\frac{1}{4}, \frac{2}{7}]$. If $s \leq 2$, then $U \leq 3r_1 < 1$; otherwise, $s > 2 \Rightarrow \sum_{i=2}^{s} w_i < 0.5 - w_1 < \frac{1}{4}$:
CASE 3.1: $w_2 \in (\frac{1}{7}, w_1]$, then $\sum_{i=3}^{s} w_i < \frac{1}{4} - \frac{1}{7} = \frac{3}{28}$.
$U < r_1 + r_2 + 2\sum_{i=3}^{s} w_i + 2w_{s+1} \leq \frac{4}{7} + 4\sum_{i=3}^{s} w_i < 1$.
CASE 3.2: $w_2 \leq \frac{1}{7}$.
CASE 3.2.1: $w_s \leq \frac{1}{14}$, then $U < r_1 + 2\sum_{i=2}^{s} w_i + 2w_{s+1} < \frac{2}{7} + \frac{1}{2} + \frac{1}{7} < 1$.
CASE 3.2.2: $w_s > \frac{1}{14}$, then $\frac{1}{7} \geq w_2 \geq w_3 \geq ... \geq w_s > \frac{1}{14}$. Since $\sum_{i=2}^{s} w_i < \frac{1}{4}$, $s \leq 4$. $U = r_1 + \sum_{i=2}^{s} r_i + r_{s+1} \leq r_1 + 4r_2 = \frac{2}{7} + 4 \cdot \frac{1}{7} < 1$.
CASE 4: $w_1 \leq \frac{1}{7}$, then (i) for $i = 1, 2, ..., s+1$, $r_i \in \{\frac{1}{7}, \frac{1}{14}, \frac{1}{28}, \frac{1}{56}, ...\}$; (ii) $r_1 \geq r_2 \geq ... \geq r_s \geq r_{s+1}$. Since $\sum_{i=1}^{s} w_i < 0.5$ and $r_i < 2w_i$, $(1 - \sum_{i=1}^{s} r_i) > 0$. By (i) and (ii), $\forall i \in [1, s]$, r_i is divisible by r_{s+1}. It follows $(1 - \sum_{i=1}^{s} r_i)$ is divisible by r_{s+1}. Therefore, $r_{s+1} \leq 1 - \sum_{i=1}^{s} r_i \Rightarrow U \leq 1$. ■

Theorem 4.3 $\Upsilon_{\textit{MulZ-FFD}} = 0.5$.

Proof: (i) Since $\{(m+1) \times (0.5 + \epsilon)\}_{rp}$ is unschedulable on an m-resource by *MulZ-FFD*, $\Upsilon_{\textit{MulZ-FFD}} \leq \frac{(m+1)\cdot(0.5+\epsilon)}{m} \to 0.5$ when $m \to \infty$ and $\epsilon \to 0$. It follows $\Upsilon_{\textit{MulZ-FFD}} \leq 0.5$. (ii) Suppose $\{w_1, w_2, ..., w_n : 1 \geq w_1 \geq w_2 \geq ... \geq w_n > 0\}_{rp}$ is unschedulable on an m-resource by *MulZ-FFD*. Find the proper t where $\{w_i : i = 1, 2, ..., t\}_{rp}$ is schedulable and $\{w_i : i = 1, 2, ..., t+1\}_{rp}$ is unschedulable. If $\sum_{i=1}^{t} w_i < 0.5$, there is a resource-unit whose utilization is less than 0.5. By Lemma 4.5, it is able to accommodate a regular partition of weight w_{t+1}. This contradicts that $\{w_i : i = 1, 2, ..., t+1\}_{rp}$ is unschedulable. Therefore, $\sum_{i=1}^{t} w_i \geq 0.5$. It follows that the total weight of any unschedulable partition set is greater than 0.5 and $\Upsilon_{\textit{MulZ-FFD}} \geq 0.5$. ■

V. Experimental Results

A. Regular Partition Scheduling on Mulitresources

Let us briefly explain this part of the simulation experiments. For each weight percentile, we generate 50000 random partition sets. In these sets, the weight of each regular partition is randomly chosen in the interval

Resource-Unit	E-ABS	Partitions	Approx. Weights
No. 0	$\mathcal{Z}_{3,2}$	0.65, 0.3	$\frac{2}{3}, \frac{1}{3}$
No. 1	$\mathcal{Z}_{5,2}$	0.6, 0.35	$\frac{3}{5}, \frac{2}{5}$
No. 2	$\mathcal{Z}_{7,2}$	0.55, 0.25	$\frac{4}{7}, \frac{2}{7}$
No. 3	$\mathcal{Z}_{4,2}$	0.5, 0.25, 0.25	$\frac{1}{2}, \frac{1}{4}, \frac{1}{4}$

TABLE IV
A Partitioning Example of *MulZ-FFD*

$\Theta = (low, high)$. Then we simulate different partitioned scheduling algorithms and count their schedulability rates.

When a single approximation sequence is applied, Figure 7 shows that $\mathcal{Z}_{7,2}$-*FFD* has the highest overall schedulability rate due to its lowest approximation overhead.

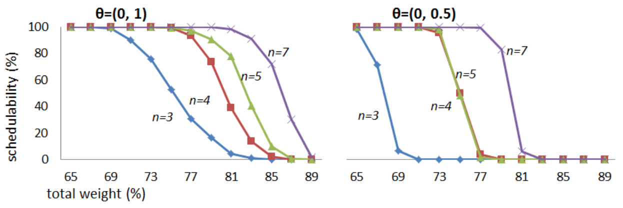


Fig. 7. Schedulability % of $\mathcal{Z}_{n,2}$-*FFD* on a 64-resource

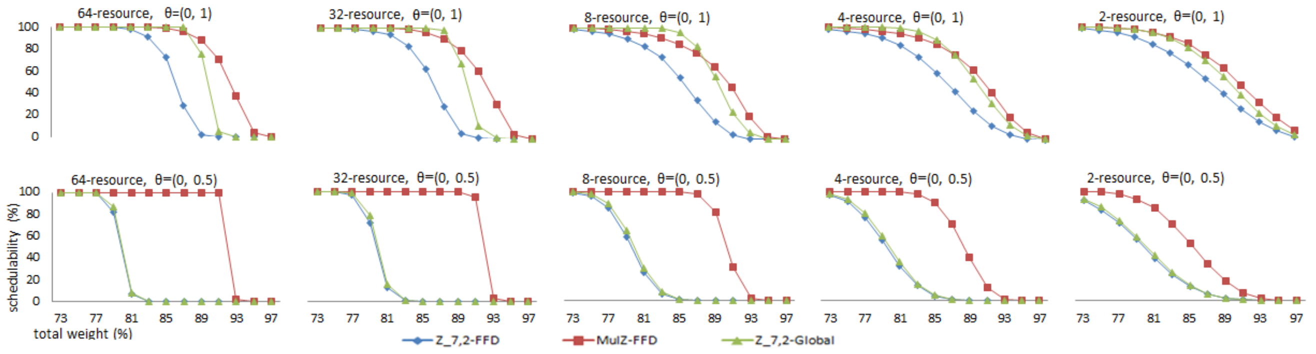


Fig. 8. *MulZ-FFD* Greatly Improves Schedulability %

Figure 8 compares the schedulability rate among *MulZ-FFD*, $\mathcal{Z}_{7,2}$-*FFD* and $\mathcal{Z}_{7,2}$-*Global*. We have shown that $\mathcal{Z}_{7,2}$-*FFD* achieves the optimal overall resource utilization with a single E-ABS. Meanwhile, $\mathcal{Z}_{7,2}$-*Global* (Magic7 [5]) is the current optimal global scheduling algorithm for regular partitions. We ignore its migration overhead because it is hard to estimate a proper value for it. This kind of overhead depends on the physical platform architecture, which is beyond the scope of this paper. Meanwhile, this absence will not impact our conclusions.

The simulation results indicate that *MulZ-FFD* has better performance than the others. First, each chart shows that *MulZ-FFD* outperforms $\mathcal{Z}_{7,2}$-*FFD*, which means *MulZ-FFD* drastically improves the overall resource utilization by using multiple approximation sequences to reduce the approximation overhead. Second, even without considering the migration overhead due to global scheduling, *MulZ-FFD* performs better or no worse than $\mathcal{Z}_{7,2}$-*Global* when $\Theta = (0, 1)$, and significantly outperforms $\mathcal{Z}_{7,2}$-*Global* for light-weight partitions where $\Theta = (0, 0.5)$. Moreover, *MulZ-FFD* performs even better on middle-to-large multiresource platforms.

B. Compare the Periodic and the Regularity-based Models

Then we compare the overall resource utilization between the Periodic and the Regularity-based Models. As shown in Table I, for task-level scheduling in the Periodic Model, we can only find solutions for the simple periodic task model. Therefore, our comparison only focuses on this task model, where a periodic task t_i is defined as (c_i, p_i). c_i and p_i are t_i's execution time and period, respectively.

At the beginning of the simulation, a size of the computation resource, m, is selected. The main body is a 10000-run loop. In each run, we generate a group of periodic task sets $\{T_0, T_1, ...T_n : T_i = \{t_{i0}, t_{i1}, ...\}\}$ for each weight percentile r, where the total weight of these task sets is exactly $m \cdot r/100$ and the weight of each task set is randomly chosen in the interval $\Theta = (low, high)$. It follows two major phases. Phase I simulates task scheduling. For each task set T_i, we determine a resource partition P_i which has the exact size to accommodate it with the EDF policy. And Phase II simulates resource partitioning. We determine the schedulability of resource partitions $P_0, P_1, ..., P_n$ on an m-resource. After the 10000-run main loop, we count the schedulability rate.

We implement our simulation in three scenarios: on a single resource, on a multiresource with global resource scheduling and on a multiresource with partitioned resource scheduling. In all of them, we apply schedulability tests in [10] and [15] for task scheduling in the Periodic and the Regularity-based Models, respectively. Figure 9 shows our experimental results on a single resource. We apply EDF and $\mathcal{Z}_{7,2}$ for resource scheduling in the two models respectively. The results show that the Regularity-based Model has higher schedulability rate most of the time. Figure 10 shows the experimental results on a 64-resource with global resource scheduling. P-fair and Magic7 are applied for resource scheduling. We find that the Regularity-based Model also outperforms the Periodic Model especially when task sets are light ($\Theta = (0, 0.5)$). Figure 11 shows the experimental results on a 64-resource with partitioned resource scheduling. *EDF-FFD* and *MulZ-FFD* are applied for resource scheduling. The results show that the Regularity-based Model outperforms the Periodic Model in both scenarios, no matter the task sets are heavy or light. In general, the Regularity-based Model achieves higher schedulability rate than the Periodic Model, which shows that it also provides higher overall resource utilization.

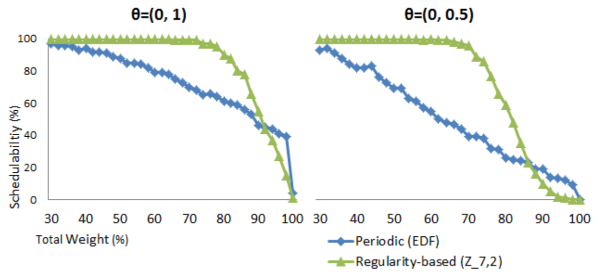


Fig. 9. Schedulability % on a Single Resource

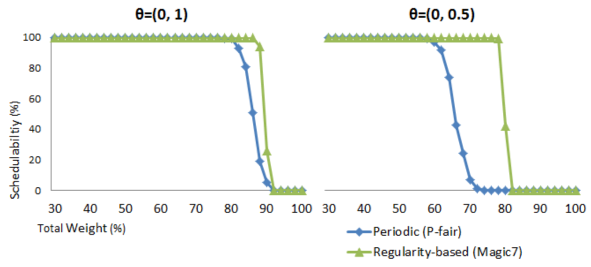


Fig. 10. Schedulability % on a 64-Resource with Global Resource Scheduling

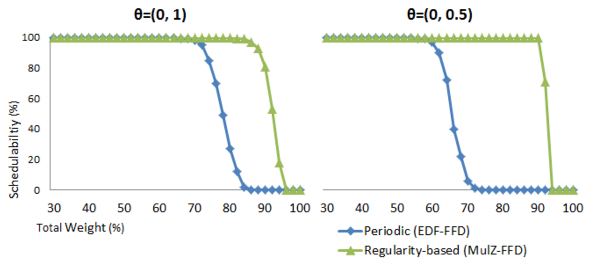


Fig. 11. Schedulability % on a 64-Resource with Partitioned Resource Scheduling

VI. Conclusion

The Periodic Model is the most popular resource model for HiRTS due to its simplicity in resource partitioning. However, it has not solved most classical task scheduling problems because of the significant blacked-out intervals on its resource partitions. The Regularity-based Model achieves maximal transparency for task scheduling, but its resource partitioning problem is complicated due to the very strict timing constraint on regular partitions. To alleviate the weaknesses of the Regularity-based Model, this paper introduces new resource partitioning techniques for it. After applying these new techniques, our simulation results show that the Regularity-based Model achieves higher overall resource utilization than the Periodic Model. Since the Regularity-based Model can also handle much more task models at the task level, we conclude that the Regularity-based Model is a better choice than the Partitioned Model for the integration of real-time applications.

ACKNOWLEDGMENT

This paper is based upon work supported in part by the National Science Foundation under Awards No. 0720856 and No. 1219082.

REFERENCES

[1] A. K. Mok and X. Feng. Towards compositionality in real-time resource partitioning based on regularity bounds. RTSS, 2001.
[2] A. K. Mok, X. Feng, and D. Chen. Resource partition for real-time systems. RTAS, 2001.
[3] X. Feng. Design of real-time virtual resource architecture for largescale embedded systems. Ph.D. dissertation, Department of Computer Science, The University of Texas at Austin, 2004.
[4] Y. Li, A. M. K. Cheng, A. K. Mok. Regularity-based partitioning of uniform resources in real-time systems. RTCSA, 2012.
[5] Y. Li, and A. M. K. Cheng. Static approximation algorithms for regularity-based resource partitioning. RTSS, 2012.
[6] J.M. Lopez, J.L. Diaz, and D.F. Garcia. Utilization bound for EDF scheduling on real-time multiprocessor systems. Real-Time Systems, 28(1):39C68, 2004.
[7] S. Shigero, M. Takashi, and H. Kei. On the schedulability conditions on partial time slots. RTCSA, 1999.
[8] S. Baruah, N. Cohen, G. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. Algorithmica, 1996.
[9] S. Baruah, J. Gehrke, and G. Plaxton. Fast scheduling of periodic tasks on multiple resources. IPPS, 1995.
[10] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. RTSS, 2003.
[11] A. Easwaran, M. Anand, and I. Lee. Compositional analysis framework using EDP resource models. RTSS, 2007.
[12] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming environment in a hard real-time environment. J. ACM, 20(1):46-61, 1973.
[13] A. Bastoni, B.B. Brandenburg and J.H. Anderson. An empirical comparison of global, partitioned, and clustered multiprocessor EDF schedulers. RTSS, 2010.
[14] A. Bastoni, B. Brandenburg, and J. Anderson. Is semi-partitioned scheduling practical? ECRTS, 2011.
[15] Y. Li and A. M. K. Cheng. Transparent Real-Time Task Scheduling on Temporal Resource Partitions. IEEE Transaction on Computers (TC), 2015.
[16] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. RTSS, 1998.
[17] Y. Li, and A. M. K. Cheng. The Value of 'Even Distribution' for Temporal Resource Partitions. Technical Report, University of Houston, 2016.
[18] Z. Deng and J. Liu. Scheduling real-time applications in an open environment. RTSS, 1997.

Appendix A

Lemma A1 shows that the time slices on a regular partition are always evenly distributed. For convenience, a span denotes two neighboring time slices on a regular partition. Meanwhile, the size of a span is the distance between the two time slices it represents.

Lemma A1 *The size of any span on a regular partition is either $\lfloor \frac{1}{w} \rfloor$ or $\lceil \frac{1}{w} \rceil$, where w is the partition weight. Both types of spans coexist if $\frac{1}{w}$ is a fraction.*

Proof: Suppose $w = \frac{q}{p}$ where p, q are co-prime. We only need to check the property on the standard regular partition P with time-slice sequence $T_0(p,q)$. By Def. 2.3, the first time slice preempted on P must be 0; otherwise, the instant regularity at time 1 is less than 0. Notice that $\forall n \in (0,q)$, $\lfloor \frac{n \cdot p}{q} \rfloor \cdot \frac{q}{p} \leq n$; $(\lfloor \frac{n \cdot p}{q} \rfloor + 1) \cdot \frac{q}{p} > n$ $\Rightarrow \lfloor \frac{n \cdot p}{q} \rfloor \cdot \frac{q}{p} \in (n - \frac{q}{p}, n]$. Consider the second time slice on P. By Lemma 2.1, a preempted time slice increases the instant regularity of P by $(1 - \frac{q}{p})$, and a non-preempted time slice decreases it by $\frac{q}{p}$. To ensure the value of every instant regularity in [0,1) (Def.s 2.2 and 2.3), the second preempted time slice must be $\lfloor \frac{p}{q} \rfloor$ because $\lfloor \frac{p}{q} \rfloor \cdot \frac{q}{p} \in (1 - \frac{q}{p}, 1]$ †. Similarly, the third time slice must be $\lfloor \frac{2p}{q} \rfloor$; ... the q-th time slice must be $\lfloor \frac{(q-1)p}{q} \rfloor$. Therefore, $T_0(p,q)$ equals $\langle 0, \lfloor \frac{p}{q} \rfloor, \lfloor \frac{2p}{q} \rfloor, ..., \lfloor \frac{(q-1)p}{q} \rfloor \rangle$. $\forall k \in (0, q]$, let $span(k)$ denote the distance between the k-th and (k+1)-th time slices in $T_0(p,q) \cup \langle p \rangle$, then $span(k) = \lfloor \frac{k \cdot p}{q} \rfloor - \lfloor \frac{(k-1) \cdot p}{q} \rfloor \in [\lfloor \frac{p}{q} \rfloor, \lfloor \frac{p}{q} \rfloor + 1]$. When $\frac{p}{q}$ is a fraction, $\exists k$, $span(k) = \lfloor \frac{p}{q} \rfloor + 1$; otherwise, the total size of all spans in $T_0(p,q) \cup \langle p \rangle$ is $q \cdot \lfloor \frac{p}{q} \rfloor < p$. ■

Appendix B

Proof of Lemma 2.2: We only need to show that if $T(p,r,\delta) \subset T_0(p,q)$ where $q + r < p$ and $r < q < 2r$, then $\exists n > 0$, $\frac{q}{p} = \frac{2n+1}{4n+3}$; $\frac{r}{p} = \frac{n+1}{4n+3}$.

Let S (resp. S') denote the infinite sequence including all time slices on the regular partition $T_0(p,q)$ (resp. $T(p,r,\delta)$), then S' is a subsequence of S. Let $d = \lfloor \frac{p}{q} \rfloor$ and $d' = \lfloor \frac{p}{r} \rfloor$. By Lemma A1, any span size in S (resp. S') is either d or $d+1$ (resp. d' or $d'+1$). Meanwhile, since $q + r < p$ and $r < q < 2r$, we have $d \leq d' \leq 2d + 1$.

CASE 1: $d \geq 3$. Since $d \leq d' \leq 2d+1$ and $2d > (d+1)+1$, the size of any span in S' can only be chosen from either $\{d, d+1\}$ or $\{2d, 2d+1\}$. The first case is impossible because there must be a pair of neighboring time slices in S' separated by a time slice in $S - S'$, the distance of which is not less than $2d$. In the second case, the time slices in S must be assigned to S' and $S - S'$ alternately. It follows $q = 2r$, which contradicts $q < 2r$.

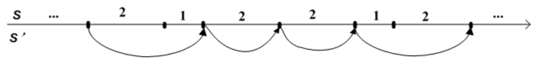
CASE 2: $d = 1$. Then $\left\lfloor \frac{p}{q} \right\rfloor = d = 1; p > q + r; r < q < 2r$

$\Rightarrow \frac{1}{2}p < q < \frac{2}{3}p$; $\frac{1}{4}p < \frac{1}{2}q < r < 1 - q < \frac{1}{2}p$

$\Rightarrow \frac{1}{2} < \frac{q}{p} < \frac{2}{3}$; $2 < \frac{p}{r} < 4$.

From $\frac{1}{2} < \frac{q}{p} < \frac{2}{3}$, there are no neighboring 1-size spans in S ‡; from $2 < \frac{p}{r} < 4$, $d' = \lfloor \frac{p}{r} \rfloor = 2$ or 3.

CASE 2.1: $d' = 2$. Suppose there are x (resp. y) times 1-size (resp. 2-size) spans in the first period of S. Then $x + 2y = p, x + y = q \Rightarrow y = p - q$.



As shown in the figure, since there are no neighboring 1-size spans in S, each 2-distance span in S' corresponds to a 2-distance one in S, and each 3-distance span in S' corresponds to a 1-distance one and a 2-distance one in S. Suppose there are x' (resp. y') times 3-size (resp. 2-size) spans in the first period of S', then $x' = x, y' = y - x \Rightarrow r = x' + y' = y \Rightarrow r = p - q$, which contradicts $q + r < p$.

† Suppose the second preempted time slice is t. If $t < \lfloor \frac{p}{q} \rfloor$, then $I_P(t+1) \geq 1$. If $t > \lfloor \frac{p}{q} \rfloor$, then $I_P(t) < 0$. Both cases contradict that $\forall t', I_P(t') \in [0,1)$.

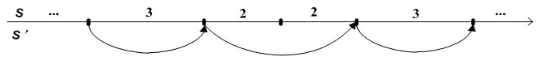
‡ Three consecutive preemptive time slices increase the instant regularity on S by $3 \cdot (1 - \frac{q}{p}) > 1$, which contradicts Def. 2.2.

CASE 2.2: If $d' = 3$, any span size in S' is 3 or 4, then each span in S' corresponds to at least two consecutive spans (size 1 or 2) in S. It follows $q \geq 2r$, which contradicts $q < 2r$.

CASE 3: $d = 2$. Then $2 \leq d' \leq 5$.

CASE 3.1: $d' = 2$ is impossible because there must be a pair of neighboring time slices in S' separated by a time slice in $S - S'$, the distance of which is not less than $2d = 4$.
CASE 3.2: If $d' = 4$ or 5, the time slices in S must be assigned to S' and $S - S'$ alternately. It follows $q = 2r$, which contradicts $q < 2r$.

CASE 3.3: If $d' = 3$, the span size in S' is 3 or 4. As shown in the figure, each 3-size (resp. 4-size) span in S' corresponds to a 3-size span (resp. two consecutive 2-distance spans) in S. Also, 4-distance spans must exist in S'; otherwise, $r = q$, which contradicts $r < q$.



Suppose there is a series of consecutive 2-size spans between two nearest 3-size spans in S, whose number is $2n > 0$, then $\frac{p}{q} \in (2 + \frac{1}{2n+2}, 2 + \frac{1}{2n})$ [§]. Thus, the number of consecutive 2-size spans between any two nearest 3-size spans in S equals $2n$. It follows that $\frac{q}{p} = \frac{2n+1}{4n+3}$ and $\frac{r}{p} = \frac{n+1}{4n+3}$. ■

APPENDIX C

From Lemma A1, we know that the distance between two neighboring time slices on a regular partition is either $\lfloor \frac{1}{w} \rfloor$ or $\lceil \frac{1}{w} \rceil$, but we are still interested in how these distances (or spans) are distributed. To show that, we define $T'(p,q)$ as a sequence containing the sizes of these spans in $T_0(p,q)$ in order. For example, $T_0(15,4) = \langle 0,3,7,11 \rangle$, then $T'(15,4) = \langle 3,4,4,4 \rangle$. Figure 12 shows the details. Due to space limitation of the figure, we use letter $'a'$ instead of number $'10'$, and so on.

Definition C1 *$T'(p,q) = \langle t_{i+1} - t_i : 0 \leq i < q \rangle$, assuming $T_0(p,q) = \langle t_i : 0 \leq i < q \rangle$ and $t_q = p$.*

We define $L(p,q)$ as a sequence containing the indexes of those long spans in $T'(p,q)$ in order. For example, $T'(15,4) = \langle 3,4,4,4 \rangle \Rightarrow L(15,4) = \langle 1,2,3 \rangle$. Since $T_0(p,q)$ has q spans totally where $(p \mod q)$ of them are long spans with size $\lfloor \frac{p}{q} \rfloor + 1$, $|L(p,q)| = p \mod q$.

Definition C2 *$L(p,q) = \langle i : t'_i = \lfloor \frac{p}{q} \rfloor + 1 \rangle$, assuming $T'(p,q) = \langle t'_i : 0 \leq i < q \rangle$.*

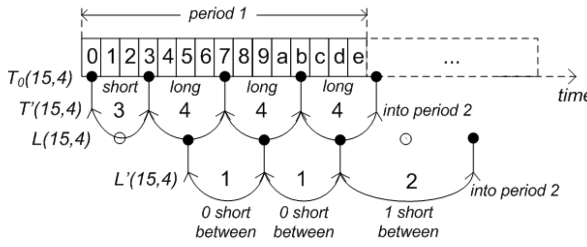


Fig. 12. Compute $L'(15,4)$ from $T_0(15,4)$

Also, we define $L'(p,q)$ to determine the difference between two neighboring elements in $L(p,q)$. For example, $L'(15,4) = \langle 1,1,2 \rangle$. By decreasing the values in $L'(p,q)$ by 1, we directly deduce the number of short spans between two neighboring long spans. Figure 12 shows the computation from $T_0(15,4)$ to $L'(15,4)$.

Definition C3 *$L'(p,q) = \langle l_{i+1} - l_i : 0 \leq i < r \rangle$, assuming $L(p,q) = \langle l_i : 0 \leq i < r \rangle$ and $l_r = q + l_0$, where $r = p \mod q$.*

Lemma C1 shows that not only $T_0(p,q)$, but also $L(p,q)$ are evenly distributed.

[§]With the following preemptive time slice, 2n times consecutive 2-size spans increase the instant regularity by $2n+1-(4n+1)\frac{q}{p} < 1$ (Def. 2.2) $\Rightarrow \frac{p}{q} < 2 + \frac{1}{2n}$. Similarly, excluding the first preemptive time slice, two times 3-size spans and 2n times 2-size spans decrease the instant regularity by $(4n+5)\frac{q}{p} - (2n+1) < 1 \Rightarrow \frac{p}{q} > 2 + \frac{1}{2n+2}$.

Lemma C1 *$\forall p, q$ where $p > q > 1$ and p,q are co-prime,* $\max\{l' : l' \in L'(p,q)\} - \min\{l' : l' \in L'(p,q)\} \leq 1$.

Proof: Let $d = \lfloor \frac{p}{q} \rfloor \geq 1$.

(I) First, let's study the case when all the elements of $L'(p,q)$ are equal to n. $n = 1$ is impossible; otherwise, all the elements of $T'(p,q)$ are equal to $(\lfloor \frac{p}{q} \rfloor + 1)$. Then $n > 1 \Rightarrow |L'(p,q)| = 1$; otherwise, $T'(p,q)$ can be divided into several identical segments, which contradicts the fact that p, q are co-prime (For example, if $L'(p,q) = \langle 2, 2\rangle$, $T'(p,q) = \langle d, d+1, d, d+1\rangle \Rightarrow p = 4d+2, q = 4$). Therefore, there is only one element of $(d+1)$ in $T'(p,q)$, together with $(n-1)$ elements of d (Def.s 3.3, 3.4). Then $q = |T'(p,q)| = n$; $p = nd+1$ (Def. 3.2) $\Rightarrow \frac{p}{q} = d + \frac{1}{n}$. Vice versa, if $\frac{p}{q} = d + \frac{1}{n}$ where $n > 1$ and p, q are co-prime, then $p = nd+1, q = n \Rightarrow T_0(p,q) = \langle 0, d, 2d, ..., (n-1)d\rangle$ (Def.s 2.2, 2.3, 2.4) $\Rightarrow T'(p,q) = \langle d, d, d, ..., d+1\rangle$ (Def. 3.2) $\Rightarrow L(p,q) = \langle n-1\rangle$ (Def. 3.3) $\Rightarrow L'(p,q) = \langle n\rangle$ (Def. 3.4).

(II) If there are elements with different values in $L'(p,q)$, $\exists n > 0$, $\frac{p}{q} \in (d+\frac{1}{n+1}, d+\frac{1}{n}) \Rightarrow \frac{q}{p} \in (\frac{n}{nd+1}, \frac{n+1}{nd+d+1})$. Assume $n+k \in L'(p,q)$ where $k \geq 2$. Consider the changes of the supply regularity in $T_0(p,q)$. A short span increases the supply regularity by $(1 - \frac{q}{p} \cdot d)$. With the following preempted time slice, $(n+k-1)$ straight short spans increase the supply regularity by: $(n+k-1)(1-d\cdot\frac{q}{p}) + (1-\frac{q}{p}) \geq (n+1)(1-d\cdot\frac{q}{p}) + (1-\frac{q}{p})$ $= n+2-(nd+d+1)\cdot\frac{q}{p} > 1$. It contradicts the definition of regular partition (Def. 2.2). Therefore, $\underline{\forall k \geq 2,\ n+k \notin L'(p,q)}$. Similarly, if $n-k \in L'(p,q)$ where $k \in [1, n)$, excluding the first preempted time slice, two neighboring long spans and the $(n-k-1)$ straight short spans between decrease the supply regularity by: $d\cdot\frac{q}{p} - (n-k-1)(1-d\cdot\frac{q}{p}) - (1-\frac{q}{p}) + d\cdot\frac{q}{p} \geq 2d\cdot\frac{q}{p} - (n-2)(1-d\cdot\frac{q}{p}) - (1-\frac{q}{p}) = (nd+1)\cdot\frac{q}{p} - (n-1) > 1$. It also contradicts the definition of regular partition. Therefore, $\underline{\forall k \in [1,n),\ n-k \notin L'(p,q)}$. The only two possible values in $L'(p,q)$ are n and $n+1$. ■

A. Schedulability Bound on a Single-resource

Then we check the following case. $\forall p, q$ and $p > q$, the time-slice sequences $T(p,q,\delta)$ $(\delta = 0, 1, ..., \lfloor \frac{p}{q} \rfloor - 1)$ can be accommodated on a 1-resource without conflict, and each long span of $T_0(p,q)$ leaves its last time slice unused in the end. In Figure 5, time slices labeled by $'6', 'a'$ and $'e'$ are left out from $T_0(15,4), T(15,4,1)$ and $T(15,4,2)$. Let $D(p,q)$ denote these remaining time slices. Obviously, $|D(p,q)| = p - q \cdot \lfloor \frac{p}{q} \rfloor = p \mod q$.

Definition C4 $D(p,q) = \langle 0, 1, ..., p-1\rangle - \bigcup_{\delta=0}^{\lfloor \frac{p}{q} \rfloor - 1} T(p,q,\delta)$.

Also, we define $D'(p,q)$ to show the distances between the neighboring time slices in $D(p,q)$. For example, $D(15,4) = \langle 6, 10, 14\rangle$, $D'(15,4) = \langle 4, 4, 7\rangle$.

Definition C5 $D'(p,q) = \langle t_{i+1} - t_i : 0 \leq i < r\rangle$, *assuming* $D(p,q) = \langle t_i : 0 \leq i < r\rangle$ *and* $t_r = t_0 + p$, *where* $r = p \mod q$.

Lemma C2 shows how to compute $D'(p,q)$ from $L'(p,q)$. Notice that they both have the size of $(p \mod q)$.

Lemma C2 *Suppose* $L'(p,q) = \langle l'_i : 0 \leq i < r\rangle$ *and* $D'(p,q) = \langle d'_i : 0 \leq i < r\rangle$ *where* $r = p \mod q$, *then* $d'_i = l'_i \cdot \lfloor \frac{p}{q} \rfloor + 1$ *for* $0 \leq i < r$.

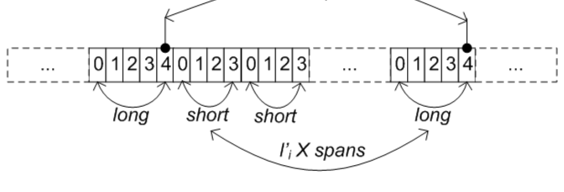
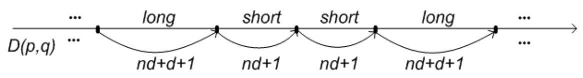


Fig. 13. Compute $D'(p,q)$ from $L'(p,q)$

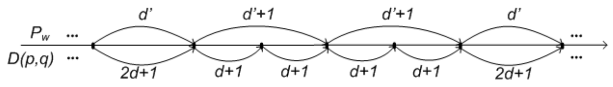
Proof: As shown in Figure 13, $D(p,q)$ is composed of the last time slices of the long spans (labeled by $'4'$ in the example). Since $(l'_i - 1)$ is the number of short spans between two neighboring long spans, $d'_i = (l'_i - 1)\cdot\lfloor \frac{p}{q} \rfloor + (\lfloor \frac{p}{q} \rfloor + 1) = l'_i \cdot \lfloor \frac{p}{q} \rfloor + 1$ for $0 \leq i < r$. ■

Lemma C3 *Suppose* $\frac{q}{p} < \frac{1}{2}$ *where p, q are co-prime, and* $r = (p \mod q) > 0$. *If* $\{\lfloor \frac{p}{q} \rfloor \times \frac{q}{p}, w\}_{rp}$ *is on-1-schedulable and* $w \in (\frac{r}{2p}, \frac{r}{p})$, *then* $p \cdot w$ *is an integer.*

Proof: Let $d = \lfloor \frac{p}{q} \rfloor$, then $d \geq 2$. By Lemma C1, there are two cases for the values in $L'(p,q)$.
CASE 1: $L'(p,q)$ only contains n. We have examined this case in the proof part (I) of Lemma C1, where we have proved $|L'(p,q)| = 1$. It follows that $|D(p,q)| = 1$ (Lemma C2), and $D(p,q)$ is the time-slice sequence of a $\frac{1}{p}$-weight regular partition. Hence, $\{\lfloor \frac{p}{q} \rfloor \times \frac{q}{p}, w\}_{rp}$ is <u>not on-1-schedulable</u> because $w \not\preceq_{rp} \frac{1}{p}$ (Lemma 2.3).
CASE 2: $L'(p,q)$ contains both n and $n+1$ where $n > 0$. By Lemma C2, $D'(p,q)$ contains both $nd+1$ and $nd+d+1$, as shown in the next figure. Let $short = nd+1$ and $long = nd+d+1$, then $long > short + 1$ and $long < 2 \times short$. Since $\{\lfloor \frac{p}{q} \rfloor \times \frac{q}{p}, w\}_{rp}$ is schedulable, suppose P_w is the w-weight regular partition,



then the time-slice sequence on P_w is a part of $D(p,q)$ within their hyper period. And because $w \in (\frac{r}{2p}, \frac{r}{p})$, P_w contains more than a half of time slices in $D(p,q)$. Let $d' = \lfloor \frac{1}{w} \rfloor$, and the size of each span on P_w is either d' or $d'+1$. Then, $d' + 1 \geq long$ * and $d' < 2 \times long$ † $\Rightarrow d' \in (short, 2 \times long)$.
CASE 2.1: $\frac{1}{w}$ is an integer. The time-slice sequence of P_w is an arithmetic one and the size of each span is d' (Lemma A1). Since $d' \in (short, 2 \times long)$ and $2 \times short > long$, the only possible values of d' are $long, 2 \times short$ and $long + short$. (i) If $d' = long$, P_w cannot go over any short span of $D(p,q)$ because $short < d' < 2 \times short$. (ii) If $d' = 2 \times short$, P_w cannot go over any long span of $D(p,q)$ because $long < d' < long + short$. (iii) If $d' = long + short$, P_w contains exactly a half of the time slices in $D(p,q)$ because it selects time slices from $D(p,q)$ alternately. It follows $w = \frac{r}{2p}$, which <u>contradicts</u> the assumption $w \in (\frac{r}{2p}, \frac{r}{p})$.
CASE 2.2: $\frac{1}{w}$ is a fraction. There are two sizes of spans on P_w: d', $d'+1$. Since $d' \in (short, 2 \times long)$, the only possible values of d' and $d'+1$ are $long, 2 \times short, long + short$, and $2 \times long$, where $long < 2 \times short < long + short < 2 \times long \Rightarrow long + 1 = 2 \times short$ OR $2 \times short + 1 = long + short$ OR $long + short + 1 = 2 \times long \Rightarrow long = 2 \times short - 1$ (because $long > short + 1$) $\Rightarrow n = 1$ (apply $short = nd + 1$ and $long = nd + d + 1$). Therefore, $short = d + 1$, $long = 2d + 1$, $d' = long$ and $d' + 1 = 2 \times short$. Also, the number of the short spans between each two neighboring long spans on $D(p,q)$ must be even. As shown in the next figure, P_w and $D(p,q)$ always keep synchronized — a short span of P_w corresponds to a long span of $D(p,q)$, and a long span of P_w corresponds to two short spans of $D(p,q)$. Since p is a period of $D(p,q)$, p is also a period of P_w. It follows <u>$p \cdot w$ is an integer</u>. ■



Proof of Lemma 3.1: From Def.s 2.7, 2.8, $\exists \frac{q_0}{p_0} \in B$, where $\frac{q_0}{p_0} < \frac{1}{2}$ and p_0, q_0 are co-prime. If $q_0 = 1$, just let $p = p_0$; otherwise, let $r_0 = p_0 \mod q_0$, then $r_0 > 0$. Since $\Upsilon_B > 0.5$, by Def. 2.10, $\exists w \in B, w \in (\frac{r_0}{2p_0}, \frac{r_0}{p_0})$. Since B is on-1-feasible, $\{\lfloor \frac{p_0}{q_0} \rfloor \times \frac{q_0}{p_0}, w\}_{rp}$ is on-1-schedulable. By Lemma C3, $p_0 \cdot w$ is an integer. Suppose $w = \frac{q_1}{p_1}$ where p_1, q_1 are co-prime. Then $p_0 \cdot \frac{q_1}{p_1}$ is an integer $\Rightarrow p_0$ is divisible by p_1. Therefore, $\frac{q_1}{p_1} < \frac{r_0}{p_0} < \frac{q_0}{p_0} \Rightarrow q_1 < \frac{p_1}{p_0} \cdot q_0 \leq q_0$. Repeat the same argument from p_1 and q_1 ..., and eventually, we can find p_n, q_n such that $\frac{q_n}{p_n} \in B$ and $q_n = 1$. Meanwhile, $\frac{q_n}{p_n} < \frac{q_0}{p_0} < 0.5 \Rightarrow \frac{1}{p_n} < 0.5 \Rightarrow p_n \geq 3$. Let $p = p_n$. ■

* If $d' + 1 < long$, P_w cannot go over any long span of $D(p,q)$.

† Because P_w contains more than a half of the time slices in $D(p,q)$, $d' \leq average\_span\_size\_of\_P_w < 2 \times average\_span\_size\_of\_D(p,q) < 2 \times long$.