Towards Compositionality in Real-Time Resource Partitioning Based on Regularity Bounds *

Aloysius K. Mok, Xiang (Alex) Feng
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
{mok,xf}@cs.utexas.edu

Abstract

In real-time resource partitioning, a shared resource is partitioned by a resource-level scheduler such that each partition is accessible only by an individual application task group. Tasks within the same task group are scheduled by an application-task-level scheduler that is specialized to the real-time requirements of the tasks in the group. An ideal goal for resource partitioning in real-time systems is to achieve a complete separation of concerns so that: (1) each task group may be executed as if it had access to its own dedicated resource, and (2) there is minimal interaction between the resource-level scheduler and the application-task-level scheduler. In [15], we introduced the notion of a real-time virtual resource which operates at a fraction of the rate of the shared physical resource and whose rate of operation varies with time but is bounded. In this paper, we discuss an approach to bound the variation of the rate of operation of a real-time virtual resource by characterizing the rate variation from both temporal and supply dimensions and by expanding on the concept of regularity that was first introduced in [19]. For the case of regular resource partitioning, we show that the utilization bounds of both fixed-priority scheduling and dynamic-priority scheduling remain unchanged from those for dedicated resources. We determine the utilization bounds for the more general case of irregular partitioning. In particular, both types of partitions can be efficiently constructed by exploiting compositionality properties vis-a-vis the regularity measure.

Keywords: Resource Partition, Open System, Utilization bounds, Real-Time Task Scheduling.

1 Introduction

The concept of open systems [5] has attracted substantial attention in the real-time systems community because (1) resource sharing is more economical than dedicated resources in small embedded systems; (2) in case of hardware failures, an open system environment would allows task groups to be relocated by coexisting with other task groups on the diminished pool of shared resources. In an open system, a physical resource is shared by different classes of applications, some hard-real-time, others soft-real-time or even non-real-time. Sharing is enforced by some kind of partitioning scheme that time-multiplexes the physical resource between the different application task groups with the goal that each application task group may be programmed as if it had dedicated access to a physical resource, i.e., without interference from other task groups due to resource sharing. Tasks within the same application task group are scheduled by an application-task-level scheduler that is specialized to the real-time requirements of the tasks in the group. Resource partitioning is implemented by a resource-level scheduler that may be time-triggered or event-triggered or both. Ideally, resource partitioning should achieve a complete separation of concerns so that: (1) each application task group sharing the physical resource may be executed as if it had exclusive access to its own dedicated resource, and (2) there is minimal interaction between the resource-level scheduler and the application-task-level scheduler.

Towards this end, we introduced in [15] the notion of a real-time virtual resource. A real-time virtual resource operates at a fraction of the rate of the shared physical resource but its rate of operation may not be uniform; the rate may vary with time but the variation is bounded. This is contrary to the systems assumption made in the seminal paper of Liu and Layland in 1973 [14] and in many subsequent real-time task models, e.g., [16], [6] where the physical re-
source is made available at a uniform rate and is accessible exclusively by the tasks in the model. In [15], the variation in the rate of operation of a real-time virtual resource is characterized by means of a delay bound $D$ that specifies the maximum extra time the task group may have to wait in order to receive its fraction of the physical resource over any time interval starting from any point in time. This way, if we know that an event $e$ will occur within $x$ time units from another event $e'$ assuming that the virtual resource operates at a uniform rate and event occurrence depends only on resource consumption (i.e., virtual time progresses uniformly), then $e$ and $e'$ will be apart by at most $x + D$ time units in real time. If infinite time-slicing is possible, each task group could have exclusive access to resource that is made available at a uniform rate that is a fraction of the actual rate of operation of the physical resource, and the delay bound would be zero.

In practice, the delay bound is unlikely to be zero since infinite time-slicing is impractical owing to context switching overhead costs and because of resource-specific constraints that may impose a lower bound on the time-slice size. For example, a communication bus cannot be infinitely time-sliced if a bus cycle must be at least as long as the signal propagation latency across the bus. In practice, there are numerous real-time systems in which the task parameters may be considered as integers because the resource is allocated in integral numbers of a system-defined time unit, the quantum size. Executions times, for example, are naturally expressible in terms of certain units of operation in systems such as ATM networks. Likewise, task periods are usually rounded up to integers for such practical purposes. In general, the larger the quantum size is, the smaller is the context switch overhead as a fraction of the resource usage and therefore higher utilization is achieved. A smaller quantum size on the other hand allows shorter deadlines to be met and therefore may yield a higher utilization bound in the schedulability analysis. The choice of quantum size is thus a trade-off between high utilization and efficiency in real-time scheduling. Henceforth in this paper, we shall assume that task timing parameters are integral.

In this paper, we shall discuss an approach to bound the variation of the rate of operation of a real-time virtual resource by expanding on the concept of regularity that was first introduced in [19]. Over any time interval $L$ starting from any point in time, the temporal regularity $D$ of a real-time virtual resource specifies the maximum amount of time a task group utilizing the resource may have to wait to either receive its fraction of the physical resource corresponding to the interval $L$, or to get to the point in the future when it will have gotten the fraction of the physical resource in the case it has already received the fraction of the physical resource. Thus, if an event $e$ will occur within $x$ time units from another event $e'$ assuming that the virtual resource operates at a uniform rate and event occurrence depends only on resource consumption (i.e., virtual time progresses uniformly), then $e$ and $e'$ will be apart by at least $x + D$ time units and at most $x + D$ time units when the virtual resource operates at a non-uniform rate in real time. Similarly, the supply regularity $S$ of a real-time virtual resource is the maximum value of the surplus/deficiency in resource supply over any time interval starting from any point in time. Thus, suppose $R$ denotes the nominal rate of operation of the real-time virtual resource, i.e., its fraction of the physical resource, the actual supply of the virtual resource in the interval between $e$ and $e'$ will be bounded by $xR - S$ and $xR + S$. In general, the two types of regularities of a real-time virtual resource provide us with quantitative measures that allow us to better deal with not only deadlines of periodic tasks but more general types of timing constraints such as jitter. We say that a resource partition is regularity-bounded if the corresponding real-time virtual resource has finite temporal or supply regularity.

One important type of regularity-bounded resource partitions is the regular partition which will be formally defined in the next section. For the Liu and Layland task model, an important property of regular partitions is that the utilization bounds for both fixed-priority scheduling and dynamic-priority scheduling of tasks running on regular partitions remain the same as if the task group were running on a dedicated resource whose supply rate is the same as that of the regular partition. Thus a task group scheduled on a regular partition with either scheduling policy cannot distinguish whether it is scheduled on a resource partition or a dedicated resource with the same rate. This is not true of a task group scheduled on an irregular partition. By using the regularity measures, we shall show how to compute utilization bounds for both fixed-priority and dynamic scheduling policies. These utilization bounds may be used for admission test in the open system environment.

The rate variation and therefore the regularity measures of a real-time virtual resource is in general a function of the scheduling policy used to allocate the shared physical resource among the task groups. An attractive property of regularity-bounded resource partitions is that they support compositionality. Suppose two subsystems $S_A$ and $S_B$ combine to form system $S_C$ according to certain composition rules $R$, subsystem $S_A$ by itself satisfies property $P_A$, and $S_B$ by itself satisfies property $P_B$. We say that the rules $R$ support compositionality with respect to the operator $F$ if the system $S_C$ satisfies a property given by $F(P_A, P_B)$. In the case of regularity-bounded partitions, we shall show that they can be constructed by simply combining regular partitions. For example, in order to construct a partition with supply regularity of 3, we can combine either a partition with supply regularity of 2 and another partition with supply regularity of 1 or by combining three regular parti-
tions, all of which have supply regularity 1. This efficient resource allocation mechanism plays an important role in the open system environment, since it will be shown that the impact of a regularity-bounded partition on the schedulability of tasks running on it depends only on the partition's regularity bound which can be incorporated into an efficient admission test. This enforces the desired separation of concerns; scheduling at the resource partition (task group) level and at the task level are isolated at run time.

We believe that the real-time virtual resource concept will play an important role in the domain of open systems. For example, resource partitioning is advocated in a system design concept arising in the avionics industry that is known as Integrated Modular Avionics (IMA) [1, 18]. In IMA, a single computer system with internal replication provides a common computing resource to numerous subsystems or functions. Currently, most avionics systems are implemented by a federated architecture whereby subsystems and functions are loosely coupled in order to minimize the fault propagation. One obvious disadvantage of the federated approach is the profligate usage of resources. The overall objective of IMA is to accomplish the same fault tolerance requirement as the federated approach and yet maximize resource utilization. A key idea of realizing this goal is to use temporal resource partitioning of the single computer system to ensure fault containment within each function which is inherent to the federated architecture that IMA aims at replacing.

Throughout this paper we shall assume that tasks are periodic and can start at any time, although the period can be interpreted as the minimum separation time in the sporadic task model without invalidating the results in this paper. We also assume that all task timing parameters have the domain of the non-negative integers and other variables are real numbers unless specified otherwise. Preemptive scheduling is assumed, i.e., a task executing on the shared resource can be interrupted at any instant in time, and its execution can be resumed later. Although a resource can be a processor, a communication bus, etc., we shall talk about a single processor as the resource to be shared.

Definition 1 A Task $T$ is defined as $(c, p)$, where $c$ is the (worst case) execution time requirement, $p$ is the period of the task.

Even though we do not specify a per-period deadline explicitly, we shall define deadlines when they are relevant to the result in this paper.

Definition 2 A Task Group $G$ is a collection of $n$ tasks that are to be scheduled on a real-time virtual resource (i.e., a resource partition), $G = \{T_i = (c_i, p_i)\}_{i=1}^n$.

Definition 3 The Utilization Factor $U(G)$ of a task group $G$ is $\sum_{i=1}^n (c_i/p_i)$

We use the term task group to emphasize its difference from the term task set in that a task set is to be scheduled on a dedicated resource while a task group is scheduled on a partition of the shared physical resource.

Due to page limit we have moved some proofs into a technical report that is accessible through FTP [17].

The rest of this paper is organized as follows. Sections 2 defines the partition model and its properties such as supply functions and regularities. Section 3 applies both fixed-priority scheduling and dynamic-priority scheduling to the model. The resource level scheduling is discussed in Section 4. We review related work in Section 5. Section 6 is the conclusion.

2 Regularity Resource Partition Model

2.1 Definition

Definition 4 A Resource Partition $\Pi$ (or a Partition) is a tuple $(\Gamma, P)$, where $\Gamma$ is a series of $N$ time slots $\{S_1, S_2, ..., S_N\}$ which are assigned to this partition and satisfy $0 \leq S_1 < S_2 < \cdots < S_N < P$ for some $N \geq 1$, and $P$ is the minimum period. Without loss of generality, we assume that there is no common divisor between $N$ and $P$.

We shall refer to the time slots where the processor is unavailable to the partition blocking time of the partition. In traditional models where resources are dedicated to a task group, there is no blocking time and we may consider this as a special case corresponding to the partition $\Pi = \{\{\emptyset\}, 1\}$.

Example 1 $\Pi_1 = \{\{1, 4, 5, 8\}, 9\}$ is a resource partition whose period is 9 with available resource time slots at time 1, 4, 5 and 8 during every period.

Note that by choosing different time points as the starting time we get different partition presentations. However since we assume tasks can be released at any time each of those partitions is equivalent to one another. Therefore, without the loss of generality we assume time 0 as the starting time for each partition.

Definition 5 The Availability Factor (or the Rate) of a resource partition $\Pi$ is $\alpha(\Pi) = N/P$ where $N$ is the number of time slots in $\Gamma$. It is a real number ranging from 0.0 to 1.0.

The availability factor of Partition $\Pi_1$ in Example 1 is $\alpha(\Pi_1) = 4/9 \approx 0.44$.

\footnote{It is called a time slot set in [19].}
2.2 Supply Function

In this section we shall define supply function and normalized execution as preliminary concepts before we introduce different types of regularities in the next section.

Definition 6 The Supply Function \( S(t) \) of a partition \( \Pi \) is equal to the total amount of time that is available to \( \Pi \) from time 0 to time \( t \).

There are several noteworthy properties about supply functions. First, according to whether the resource is blocked or available for each time slot \([t-1,t]\), the corresponding supply function increases or remains the same respectively. Second, given a partition \( \Pi \) with rate of \( N/P \) the earliest possible time slot assignment is \((0,1,2,\ldots,N-1)\) and the latest possible assignment is \((P-N, P-N+1, \ldots, P-1)\). Any other assignment dwells between these two extreme cases. Hence, on a graph, the supply function of Partition \( \Pi \) with rate of \( N/P \) may consist of lines on the grid of \( y=x, y=x-1, y=x-2,\ldots, y=x-N+1 \) and \( y=0, y=1,\ldots, y=N \) in the region that is encompassed by \( y=x, y=x-N-1, y=0 \) and \( y=N \). We call the lines the supply function grid of \( \Pi \). Figure 1 shows the supply function and the grid of \( \Pi_1 \) in Example 1. Though \( S(t) \) is a discrete function, it is drawn as a continuous one in order to better visualize the concept.

![Figure 1](image)

**Figure 1.** Supply Function Grid and Instant Regularity of Example 1

Definition 7 Let \( b \) denote the execution rate of the resource where partition \( \Pi \) is implemented. The Normalized Execution of partition \( \Pi \) is an allocation of resource time to \( \Pi \) at a uniform, uninterrupted rate of \((\alpha(\Pi) \times h)\). Its supply function is called normalized supply function.

The normalized execution of a partition for a task group is the execution of the task group as if it were scheduled on a dedicated resource with the same execution rate as the partition. Though it is an ideal case it is impractical due to context switching overhead costs and possible resource-specific constraints. We use it here only as a reference to measure partitions.

2.3 Regularity

In this section we shall introduce our measurements of partitions. We shall first define instant regularity and later temporal regularity and supply regularity.

Definition 8 The Instant Regularity \( I(t) \) at time \( t \) on partition \( \Pi \) is given by \( S(t) - ta(\Pi) \).

The notion of instant regularity is called dynamic regularity by Shigero et al. in [19]. We prefer the term instant regularity because it better captures the idea that it pertains to a particular time point and there are different types of regularity. The instant regularity measures at a particular time point the difference between the amount of supply that the partition has already gotten from the resource scheduler since time 0 and the amount that the partition would get if it were following its normalized execution. As shown in Figure 1 the instant regularity at time \( t \) is equal to the distance from the supply function to the normalized function. Notice that it is a real number and could be negative as well.

Definition 9 [19] Let \( a,b,e,k \) be non-negative integers, the Temporal Regularity \( R_T(\Pi) \) of Partition \( \Pi \) is equal to the minimum value of \( k \) such that \( \forall a, \forall b, 0 \leq 3e \leq k, |I(b-e) - I(a)| < 1 \).

The temporal regularity measures the overall difference between the partition supply function and the normalized supply function from the time dimension. This measurement considers all possible time intervals no matter which time point is the start point.

Definition 10 Let \( a,b,k \) be non-negative integers, the Supply Regularity \( R_S(\Pi) \) of Partition \( \Pi \) is equal to the minimum value of \( k \) such that \( \forall a, \forall b, 0 \leq k, |I(b) - I(a)| < k \).

The supply regularity is the upper bound on the amount by which the actual supply during any time interval is more than or less than the amount of supply that the partition is supposed to obtain.

These two regularities are actually interchangeable.

Corollary 1 A partition with temporal regularity of \( k \) has supply regularity of \( 1 + \frac{k}{2} \).

Corollary 2 A partition with supply regularity of \( k \) has temporal regularity of \( \left(\left(1 - \frac{1}{k}\right)\right) \).

While the temporal regularity is an integer, the supply regularity can be a real number. Temporal regularity is more suitable for task-level scheduling due to its real-time nature while supply regularity is more suitable for resource-level scheduling as we shall see in later section. The interchangeability of these two types of regularities bridges the two levels of scheduling and keeps each level focused on its own concerns.
Definition 11 [19] A Regular Partition is a partition with temporal regularity of 0.

By Corollary 1 a partition with supply regularity \( R_S(\Pi) \leq 1 \) is also a regular partition.

Definition 12 A k-Temporal-Irrregular Partition is a partition with temporal regularity of \( k \) where \( k > 0 \). A k Supply-Irrregular Partition is a partition with supply regularity of \( k \) where \( k > 1 \).

3 Task Level Scheduling

Given a partition, we now analyze the schedulability problem of a task group that may execute only during the available time slots of the partition. We shall first investigate the problem of regular partitions and later of irregular partitions.

3.1 Regular Partition

In this section, we shall show that regular partitions preserve the utilization bounds of both fixed priority scheduling (rate monotonic) and dynamic priority scheduling (earliest deadline first).

3.1.1 Fixed Priority

The computation for a utilization bound leads us to consider the concept of critical instances in partition environment. This problem was studied by Mok et al.[15] for the dense time model. Similar results apply to the integer time domain as well. In this section, we first recall some of results from [15], and then we compute the critical partitions and finally the utilization bounds on regular partitions.

Definition 13 [15] The Least Supply Function (LSF) \( S^*(t) \) of a resource partition \( \Pi \) is the minimum of \( (S(t+d) - S(d)) \) where \( t, d \geq 0 \).

\( S^*(t) \) is the smallest amount of resource time available to a partition in any interval of length \( t \).

Definition 14 [15] The Critical Partition of a resource partition \( \Pi = (\Gamma, P) \) is \( \Pi^* = (\Gamma^*, P) \) where \( \Gamma^* \) has time pairs corresponding to the steps in \( S^*(t) \) such that \( \Pi^* \)'s supply function equals \( S^*(t) \) in \((0, P)\).

Theorem 1 [15] A task group \( \tau \) is feasible in a partition \( \Pi \) if and only if it is feasible in its critical partition \( \Pi^* \).

Lemma 1 The critical partition \( \Pi^* \) of a regular partition \( \Pi \) is a partition with supply function \( S^*(t) = [\alpha(\Pi)t] \).

Proof: Let \( S(t) \) denote the supply function of \( \Pi \). To prove this lemma, we only need to prove that the least supply function of \( \Pi \) is \([\alpha(\Pi)t] \), i.e., \( \min(S(b) - S(a)) = [\alpha(\Pi)(b - a)] \) for \( \forall a \geq 0, \forall b \geq a \). From the definition of regular partitions, we have

\[
|I(b) - I(a)| < 1
\]

\[
[\alpha(\Pi) \times (b - a)] \leq S(b) - S(a) \leq [\alpha(\Pi) \times (b - a)]
\]

Since \( S(b) - S(a) \) is also an integer, it may be equal to either \([\alpha(\Pi) \times (b - a)]\) or \([\alpha(\Pi) \times (b - a)]\). We proceed with a proof by contradiction and suppose there exists \( t \) so that

\[
\min(S(b + t) - S(b)) \neq [\alpha(\Pi)t]
\]

Hence \( S(b + t) - S(b) = [\alpha(\Pi)(t)] \) for \( \forall b \geq 0 \) and \([\alpha(\Pi)(t)] \neq [\alpha(\Pi)(t)] \) (1)

Then

\[
S(b + 2t) - S(b) = S(b + 2t) - S(b + t) + S(b + t) - S(b) = 2[\alpha(\Pi)t]
\]

For the same reason, for any integer \( n \),

\[
S(b + nt) - S(b) = n[\alpha(\Pi)t]
\]

We also have

\[
S(b + nt) - S(b) \leq [\alpha(\Pi)(nt)]
\]

Therefore,

\[
n[\alpha(\Pi)t] \leq [\alpha(\Pi)(nt)]
\]

\[
[\alpha(\Pi)t] = [\alpha(\Pi)(t)]
\]

This contradicts with (1). Therefore,

\[
S^*(b - a) = \min(S(b) - S(a)) = [\alpha(\Pi)(b - a)]
\]

Lemma 2 A task group \( G \) with two tasks is schedulable on a regular partition \( \Pi \) by rate monotonic scheduling if \( U(G) \leq 2(2^k - 1)\alpha(\Pi) \).

Proof: We shall prove the lemma holds on the critical partition of Partition \( \Pi \). Similar to the proof in the seminal Liu & Layland paper [14], let \( \tau_1 \) and \( \tau_2 \) be two tasks with periods equal to \( T_1 \) and \( T_2 \) and their worst case execution times \( C_1 \) and \( C_2 \), respectively. Assume that \( T_2 > T_1 \) so \( \tau_1 \)
has higher priority than \( \tau_2 \). The minimum of the utilization factor \( U \) occurs when

\[
C_1 = S(T_2) - S(T_1 \times \left\lceil \frac{T_2}{T_1} \right\rceil) = \left[ \alpha(\Pi) \times T_2 \right] - \left[ \alpha(\Pi) \times T_1 \times \left\lceil \frac{T_2}{T_1} \right\rceil \right]
\]

and

\[
C_2 = S(T_2) - C_1 \times \left\lceil \frac{T_2}{T_1} \right\rceil = \left[ \alpha(\Pi) \times T_2 \right] - C_1 \times \left\lceil \frac{T_2}{T_1} \right\rceil
\]

Let \( \alpha(\Pi) = \frac{p}{n}, T_1 = \frac{b_2p+a_1}{n} \) where \( 0 \leq a_1 < p, T_2 = \frac{b_2p+a_2}{n} \) where \( 0 \leq a_2 < p \).

Because \( \frac{C_1}{T_1} < \frac{n}{p}, T_1n > C_1p \)

also because \( T_1n = b_1p + a_1 \)

\[
\frac{b_1p + a_1}{T_1} > C_1p
\]

Since \( C_1 \geq 1, b_1 \geq 1 \). For the same reason, \( b_2 \geq 1 \).

\[
C_1 = \left\lfloor \frac{nT_2}{P} \right\rfloor - \left\lfloor \frac{nT_1}{P} \right\rfloor = b_2 - b_1
\]

Similarly,

\[
C_2 = b_2 - (b_2 - b_1) \times 2 = 2b_1 - b_2
\]

Therefore

\[
\bullet \quad b_2 - b_1 \geq 1
\]

\[
\bullet \quad 2b_1 - b_2 \geq 1
\]

Now we compute the utilization factor:

\[
U = \frac{C_1 + C_2}{P_1 + P_2} = \frac{\left\lfloor \frac{nT_2}{P} \right\rfloor - \left\lfloor \frac{nT_1}{P} \right\rfloor}{b_2p + a_1} + 2 \frac{\left\lfloor \frac{nT_2}{P} \right\rfloor - \left\lfloor \frac{nT_1}{P} \right\rfloor}{b_2p + a_2}
\]

\[
= \frac{(b_2 - b_1)}{b_1p + a_1} + \frac{2b_1 - b_2}{b_2p + a_2} \times n
\]

\[
\geq \frac{(b_2 - b_1)}{b_1 + 1} + \frac{2b_1 - b_2}{b_2 + 1} \times \frac{n}{p}
\]

Thus the problem becomes to another problem of solving the minimum value of

\[
\left( \frac{b_2 - b_1}{b_1 + 1} + \frac{(2b_1 - b_2)}{b_2 + 1} \right)
\]

where \( b_1, b_2 \geq 1, b_2 - b_1 \geq 1 \) and \( 2b_1 - b_2 \geq 1 \).

Since the domain of task timing parameters are integral, the execution time of a task can be increased only by an integer. Instead of computing the maximum utilization factor for schedulable task groups in the worst scenario, we may compute the maximum utilization factor for unschedulable task groups in the same scenario in order to compute the utilization bound, i.e., instead of computing the maximum of

\[
\left( \frac{b_2 - b_1}{b_1 + 1} + \frac{(2b_1 - b_2)}{b_2 + 1} \right)
\]

we compute the minimum of

\[
\left( \frac{(b_2 - b_1 + 1)}{b_1 + 1} + \frac{(2b_1 - b_2)}{b_2 + 1} \right), \left( \frac{(b_2 - b_1)}{b_1 + 1} + \frac{(2b_1 - b_2)}{b_2 + 1} \right)
\]

by increasing the execution time of either \( \tau_1 \) or \( \tau_2 \) by 1.

Let \( x = b_1 + 1, y = b_2 + 1 \). Because \( b_2 > b_1 \)

\[
(2) = \left( \frac{b_2 - b_1}{b_1 + 1} + \frac{(2b_1 - b_2 + 1)}{b_2 + 1} \right)
\]

\[
= \left( \frac{(y - x)}{x} + \frac{(2x - y)}{y} \right)
\]

\[
= \frac{x}{y} + \frac{2x}{y} - 2
\]

\[
\geq 2(2^{\frac{1}{2}} - 1)
\]

The equality is reached when \( y = \sqrt{2x} \).

**Lemma 3** A task group \( G \) of \( m \) tasks with the restriction that the ratio between any two request periods is less than \( \frac{m \leq n(2^{\frac{1}{2}} - 1)\alpha(\Pi)}{2} \) is schedulable on a regular partition \( \Pi \) or rate monotonic scheduling if \( \alpha(G) \leq m(2^{\frac{1}{2}} - 1)\alpha(\Pi) \).

**Theorem 2** A task group \( G \) of \( m \) tasks is schedulable on a regular partition \( \Pi \) or rate monotonic scheduling if \( \alpha(G) \leq m(2^{\frac{1}{2}} - 1)\alpha(\Pi) \).

This result is exactly the same as the Liu and Layland's bound for \( m \) tasks on a dedicated resource which means the utilization bound is not affected by resource partitioning at all for regular partitions.

3.1.2 Dynamic Priority

After examining the utilization bound of fixed-priority scheduling for regular partitions above, we shall now give two lemmas and then show that the utilization bound of dynamic-priority scheduling also remains the same for regular partitions.

**Theorem 3** [19] A task group \( G \) with \( n \) periodic tasks, \( \tau_1, \tau_2, \ldots, \tau_n \) is schedulable on a regular partition \( \Pi \) by the earliest-deadline-first policy if and only if \( \alpha(G) \leq \alpha(\Pi) \).
Intuitively, a task group is feasible when its least demand during any time interval $t$ is always no greater than the supply during the same time interval. We shall visualize this on the supply graph. First, because the least demand is no greater than $U(G)t$ and $U(G) \leq \alpha(\Pi)$ the demand function is no greater than the normalized supply function. Second, because the demand is an integer the demand is at most the largest integer number below the normalized supply function, i.e., the supply function of the critical partition. Therefore, the least demand is no greater than the supply, thus guaranteeing the schedulability of the task group.

3.2 Irregular Partitions

**Definition 15** The Virtual Time $V(t)$ of a partition $\Pi$ is equal to $[S(t)/\alpha(\Pi)]$ where $S(t)$ is the supply function of $\Pi$.

**Definition 16** Virtual Time Scheduling on a partition $\Pi$ is a scheduling constraint such that a job is eligible to run only when it is released and its release time is no less than the current virtual time of $\Pi$.

Virtual Time Scheduling may apply to all of the scheduling algorithms designed for dedicated resources. Therefore, we have Virtual Time Rate Monotonic Scheduling, Virtual Time Earliest-Deadline-First Scheduling, and other virtual time scheduling algorithms.

**Theorem 4** A task group $G \{T_i = (c_i, p_i)\}_{i=1}^n$ is schedulable on a k-temporal-irregular partition $\Pi$ by virtual time rate monotonic scheduling if $\sum_{i=1}^n \frac{c_i}{p_i-k} \leq \alpha(\Pi)n(2^k-1)$.

**Proof Sketch:** We construct a task group $G'$ as $c_i, p_i$ but with deadline of $p_i - k$ for each task. $G'$ is schedulable on a regular partition $\Pi'$ with $\alpha(\Pi') = \alpha(\Pi)$ using Deadline Monotonic scheduling algorithm[11]. Let us then schedule $G'$ on $\Pi$ using virtual time rate monotonic scheduling algorithm (which has the same priority order as Deadline Monotonic in this case). A job $J'$ of Task $T_i$ released at $t_1$ and finished before $\tau_1 + p_i - k$ when scheduled on $\Pi'$ will be scheduled between $t_1$ and $t_1 + p_i$ when scheduled on $\Pi$. Therefore, $G$ is schedulable.

Similarly, we have

**Theorem 5** A task group $G \{T_i = (c_i, p_i)\}_{i=1}^n$ is schedulable on a k-temporal-irregular partition $\Pi$ by virtual-time earliest-deadline-first if $\sum_{i=1}^n \frac{c_i}{p_i-k} \leq \alpha(\Pi)$.

Since earliest-deadline-first is an optimal scheduling algorithm we also have

**Theorem 6** A task group $G \{T_i = (c_i, p_i)\}_{i=1}^n$ is schedulable on a k-temporal-irregular partition $\Pi$ by earliest-deadline-first if $\sum_{i=1}^n \frac{c_i}{p_i-k} \leq \alpha(\Pi)$.

4 Resource Level Scheduling

Given the resource requirement of the availability factor and regularities for each partition $\Pi_i$, a schedule must be constructed at the resource level. In this section, we shall discuss how to schedule both regular and irregular partitions on resources.

4.1 Regular Partition

**Theorem 7** [19] A regular partition is uniquely determined by its availability factor except for the offset.

**Theorem 8** Given a set $\{n_i/p_i, 1 \leq i \leq m\}$ as the availability factors of regular partitions, the decision problem of whether there exists a schedule containing all the partitions is NP-hard.

Though the problem is NP-hard, we can always convert the availability factors into a schedulable set as long as the new availability factor is bigger than the old one. This method is similar to one solution of the pinwheel problem [6, 7].

**Theorem 9** Regular partitions whose availability factors are all powers of some number and whose total availability factor $\leq 1.0$ are schedulable.

**Example 2** Regular partitions $\Pi_i, \{1 \leq i \leq 4\}$ with availability factors of $1/2, 1/4, 1/8, 1/8$ respectively can be easily scheduled on a dedicated resource with the period of 8 and the time slot assignment of $\{1, 2, 1, 3, 1, 2, 1, 4\}$ where $i$ indicates $\Pi_i$.

**Theorem 10** Given a set $\{n_i/p_i, 1 \leq i \leq m\}$ as the availability factors of regular partitions, they are schedulable if $\sum_{i=1}^m \frac{n_i}{p_i} \leq 0.5$.

**Proof:** For each $n_i/p_i$, let $B_i = \frac{1}{2^i}$ where $\frac{1}{2^i} \geq \frac{n_i}{p_i} > \frac{1}{2^{i+1}}$. We have $B_i < 2 \times \frac{n_i}{p_i}$, therefore $\sum_{i=1}^m B_i < 1.0$. Because $B_i$ consists solely of powers of the same base of 2 it is schedulable, hence the set of partitions is also schedulable.

A better bound is possible if we use the double-integer reduction technique [5] or if we focus on certain special cases. We shall not cover these cases in this paper.

4.2 Irregular Partitions

**Theorem 11** When two regular partitions $\Pi_1$ and $\Pi_2$ from the same resource are combined together they form a new partition $\Pi_3$ with supply regularity of 2.
The reason why we require that the two regular partitions are from the same resource is to ensure that they cannot possibly have a conflicting time slot with each other. Such conflicts may occur if the partitions are necessarily from different resources such as would be the case in a distributed environment. We shall not address this issue any further in this paper.

**Example 3** When two regular partitions \( \Pi_1 \) and \( \Pi_2 \) in Example 2 are combined together, a new partition with availability factor of \( 5/8 \) and supply regularity of \( 2 \) will be generated.

**Theorem 12** When \( k \) regular partitions are combined together they form a partition with supply regularity of \( k \).

**Theorem 13** Given a set \( \{a_k, 1 \leq k \leq n\} \) as the availability factors of \( k \)-supply-irregular partitions, they are schedulable if \( \sum_{k=1}^{n} a_k \leq 0.75 \).

**Proof:** Let us rewrite each \( a_k \) as

\[
a_k = \frac{1}{2^i} + \frac{x}{2^j}
\]

where \( i = \left[ \log_2 a_k \right], j = \left[ \log_2 (a_k - \frac{1}{2^i}) \right] \) and \( x = \frac{(a_k - \frac{1}{2^i})}{2^j} \) when \( (a_k - \frac{1}{2^i}) \neq 0; j = i + 1 \) and \( x = 0 \) when \( (a_k - \frac{1}{2^i}) = 0 \).

Hence when \( x \neq 0, 1.0 \leq x < 2.0 \) and \( i < j \).

We construct \( b_k = \frac{1}{2^i} + \frac{1}{2^j} \) for each \( a_k \). It is easy to see that \( b_k \geq a_k \). Therefore, if \( b_k \) is schedulable \( a_k \) is also schedulable.

\[
\frac{a_k}{b_k} = \frac{\frac{1}{2^i} + \frac{x}{2^j}}{\frac{1}{2^i} + \frac{1}{2^j}} = \frac{2^{j-i} + x}{2^{j-i} + 2} \geq \frac{2^{j-i} + 1}{2^{j-i} + 2} = \frac{2 + 1}{2 + 2} = 0.75
\]

Therefore, \( \sum_{k=1}^{n} a_k \leq 0.75 \) then \( \sum_{k=1}^{n} b_k < 1 \). ■

**Example 4** Let us consider the scheduling of three \( 2 \)-supply-irregular partitions \( \Pi_1, \Pi_2 \) and \( \Pi_3 \) with availability factors 0.36, 0.29, 0.08 as their availability factors respectively. First, according to Theorem 13, 0.36 + 0.29 + 0.08 = 0.73 < 0.75, therefore, they are schedulable. Second, as how to construct the schedule, because \( 1/4 + 1/16 < 0.36 < 1/4 + 1/8 \) we assign two regular partitions with availability factors of \( 1/4 \) and \( 1/8 \) to \( \Pi_1 \). For the similar reasons we assign \( 1/4 \) and \( 1/16 \) to \( \Pi_2 \), \( 1/16 \) and \( 1/32 \) to \( \Pi_3 \). It is easy to show that the total availability factor does not exceed 1. Hence, a valid schedule is constructed.

**Theorem 14** Given a set \( \{a_i, 1 \leq i \leq n\} \) as the availability factors of \( k + 1 \)-supply-irregular partitions, they are schedulable if \( \sum_{i=1}^{n} a_i \leq 1 - \frac{1}{2^k+1} \).

Notice that both Theorem 13 and Theorem 14 only provide sufficient conditions for schedulability. Actually, as long as the total sum of the availability factors that are assigned to each partition does not exceed 1, these partitions are schedulable. Consider the \( \Pi_3 \) in Example 4. If we raise its requested availability factor 0.08 to as high as 0.31 which results in the total availability factor to be 0.96, the partitions are still schedulable. Here we only show the worst-case bound even though it is too pessimistic.

## 5 Related Work

In [15], the concept of a real-time virtual resource was introduced which operates at a fraction of the rate of the shared physical resource and whose rate of operation varies with time but is bounded. The focus of [15] is mainly on task-level schedulability with respect to given partitions. This paper is a companion to [15] in that we discuss in detail resource-level schedulability by making use of a specific approach to partitioning: regular and \( k \)-irregular partitions.

The concept of regularity was first introduced by Shigero, Takashi and Kei in [19] where they proved the feasibility of task groups whose utilization is no bigger than the availability factor of the regular partition the task group executes on. In this paper, we make the more general observation that the schedulability bound results in the Liu and Layland in fact all hold with regular partitions. We also generalize the regularity concept to include both temporal and supply regularities and give a number of compositional results.

The real-time virtual resource concept addresses some of the crucial issues of the open system environment. As such, this paper is related to the work pioneered by Deng and Liu [5]. In the open system environment, the admission test on a real-time task needs to be independent of any other task in the system and a global schedulability analysis is out of the question. This concept was first discussed based on an EDF kernel scheduler and was later extended to the fixed priority scheduler as kernel scheduler[9]. It was further extended to parallel and distributed systems [8]. Because in an open real-time environment the parameters of real-time tasks are no longer required to be known a priori, efficient online scheduling algorithms are needed [2, 21]. Also needed are practical mechanisms to provide isolation among tasks. One interesting approach is to assign each task a server with certain parameters [3, 12]. However, the interaction between tasks and the higher-level scheduler may increase the unpredictability in task execution and hence make other requirements such as output jitter bound difficult to realize. The main difference between our approach and previous approaches is that we minimize the
interaction between the resource-level scheduler and the application-task-level scheduler to a simple interface. Unlike previous approaches, our resource-level scheduler does not require knowledge of the task-level deadlines or their derivatives in partition scheduling. In the other direction, the task-level scheduler may need to know at most the regularity bound of the partition it executes on. More importantly, the related delay bound of the partition allows the application task scheduler to determine not only compliance with deadline requirements but also event-separation types of constraints. If the application task groups are not all specified in one common task model such as Liu and Layland periodic tasks, our partition model can still be used. We only need to figure out the schedulability conditions of the new system model on partitions. The effect of the partition scheduling on task group scheduling is captured by the partition regularity in this paper.

Recently, [13] proposed a framework for achieving inter-application isolation. In [13] an PShED (Processor Sharing with Earliest Deadlines First) algorithm is used to schedule partitions (called servers in [13]) in order to isolate the problem of scheduling tasks within a partition. This approach has the nice property that the task scheduling within a partition is the same as traditional task scheduling. However, this property totally depends on the PShED algorithm, which is the only way a partition could be scheduled. In addition, the dynamic deadlines of each partition, which is used to decide which partition to schedule, depends on the particularity of the tasks within the partition. Because of the inter-relations between how the partitions are scheduled and how the tasks are scheduled within a partition, jitter is hard to analyzed in this approach. Finally, this approach may not be able to handle resource-specific constraints such as the rigidity of time slots in a communication bus.

To highlight the better predictability of the partition model, consider the following example.

![Timing Diagram](image)

**Figure 2.** Comparison of Deng and Liu's model and this paper's model

Suppose tasks $T_1 = (1, 12)$ and $T_2 = (1, 4)$ are scheduled using the earliest-deadline-first scheduler on a partition with capacity of 1/2. As shown in Figure 2, in the model of [5] when other partitions are fully loaded the longest response time for $T_1$ could get close to 12. Since the shortest response time could be only 1 the execution consistency of $T_2$ is not desirable which might be crucial for some tasks with tight jitter requirements. In contrast the regularity-bounded partition model bounds the relative delay of partitions comparing with traditional environment. Hence, the two tasks could be assigned to a partition so that the longest response time of $T_1$ is bounded by a value that is sufficiently close to 4 which is the longest response time of $T_1$ as they are running on a virtual CPU with 1/2 speed of the original one. Once the parameters of a partition are determined the longest response time of a task inside this partition will be affected by only other tasks in the same partition. Hence better isolation among partitions is accomplished.

Finally, our work also differentiates from Proportional Share in [20]. The lag in Proportional Share holds only for intervals starting from the same time point while in this paper the partition regularity applies to any interval regardless of the starting point. This difference is crucial because: first, the partition regularities are most useful for bounding the separation between event pairs; second, in open system environments, tasks could join and exit the task group dynamically, hence measuring the supply of intervals with arbitrary start times is a necessity.

Compared with application architectural concepts such as IMA, the work in this paper provides the scheduling-theoretic foundation for those architectures. It has been pointed out that there are some significant issues that remain unsolved in the resource partition problem. First, IMA was found to have a large amount of output jitter [1]. Because the available time of a partition cannot in general be evenly distributed the completion time of a certain job of a task is affected not only by outstanding jobs of other tasks but also by the fluctuation of the partition. Second, IMA has been considered only for usage with STSPP [10, 15]. In STSPP partitions have only one continuous time slot within each period and this need not be the case. The results in this paper provide answers to some of these issues.

6 Conclusion

In this paper, we investigate the regularity-bounded resource partition model. We characterize the rate variation of resource partitions from both temporal and supply dimensions as temporal regularity and supply regularity respectively. We analyze regularity-bounded partitions with respect to the schedulability of both the fixed priority scheduling and dynamic priority scheduling. We also discuss the resource level scheduling of partitions by deriving a number of results that pertain to the construction of regularity-bounded partitions from regular partitions. All these tech-
Techniques are important for achieving compositionality which is a necessity in the open system environment where task groups may join and leave at any time, and hence the resource partitioning scheme must support efficient rules for determining the delay bounds of partitions that get created and combined on the fly.

Highlights of the results in this paper are:

1. We characterize resource partitions with both temporal regularity and supply regularity. Task-level scheduling can be easily performed with the knowledge of temporal regularity and resource-level scheduling with the knowledge of supply regularity.

2. The utilization bounds of both fixed and dynamic priority scheduling algorithms for regular partitions remain the same as for dedicated resources. The task groups scheduled on regular partitions will not be able to distinguish whether they are executed on resource partitions or on dedicated resources.

3. Resource-level scheduling solutions can be efficiently constructed owing to the compositionality properties of regularity-bounded partitions. These techniques are especially suitable for online scheduling and highly desirable for open systems.

4. Guaranteed jitter can be achieved with the regularity-bounded partition model.

There are still many open issues to be investigated. For example, the scheduling of regular partitions could be improved by approaches similar to those used in the pinwheel problem. Our jitter bounds are not exact and tighter bounds may be possible with more sophisticated partitioning schemes.

References


