Deferrable Server

line 1: Muhammad Farid Izwan Bin Mohamad Shabri  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

*Abstract*—This paper introduces Deferrable Server algorithm which is to improve aperiodic response time. It assigns higher priority aperiodic task to be completed by delaying the completion time of periodic task, so that the deadline for aperiodic tasks are met. Real-time system has both aperiodic and periodic tasks This paper also explains how deferrable server give impacts to aperiodic jobs compared to polling server. Since aperiodic tasks can arrive at any time, deferrable server is one of many ways to optimize a schedule with aperiodic task.

# Introduction (*Heading 1*)

In the 21st century, the world full of advanced technology, people keep doing new inventions and innovations, so that every job or task will be eased to handle. In real time systems, an integrated and consistent approach is required to schedule both hard and soft periodic task as well as aperiodic task. The challenge is to schedule the periodic and aperiodic activities together so that all of the tasks timing requirements are satisfied. The scheduling algorithm is created and applied either to periodic tasks or aperiodic tasks but not for both.

In the presence of periodic tasks, the easiest way to handle a group of soft aperiodic activities is to schedule them in the background; that is, when there are no periodic instances available to execute. The main disadvantage of this strategy is that for large periodic loads, the response time of aperiodic requests might be too long for some applications. As a result, background scheduling should be used only when aperiodic tasks do not have strict time requirements and the periodic load is low. The major advantage of background scheduling is its simplicity. To implement the background scheduling system, two queues are required: one (with a higher priority) for periodic tasks and one (with a lower priority) for aperiodic requests. The two queueing techniques are unrelated and can be implemented using various algorithms, such as Rate Monotonic for periodic activities and First Come First Serve for aperiodic requests. Only when the periodic queue is empty, tasks are accepted from the aperiodic queue. When a new periodic instance is activated, all aperiodic tasks are instantly preempted. The average response time of aperiodic tasks can be improved in terms of background scheduling by using a server; that is, a periodic job whose objective is to service aperiodic requests as quickly as feasible.

This paper describes Deferrable Server algorithm that can improve aperiodic time response performance during scheduling process. Lehoczky, Sha, and Strosnider [LSS87, SLS95] proposed the Deferrable Server (DS) method to reduce the average response time of aperiodic queries as compared to polling services. Same as Polling Server, Deferrable Server algorithm produces a periodic job (typically with a high priority) to service aperiodic queries. Unlike polling, however, Deferrable Server keeps its capacity if no requests are queued when the server is started. As long as the capacity has not been depleted, aperiodic requests can be served at the same server's priority. The capacity is renewed at full value at the start of each server term.

This paper is organized as follows. Section II describes overview of deferrable server. Section III gives examples of fixed priority server including Deferrable Server. Section IV explains more about Deferrable Server including rule states, model , calculation of utilization the least upper bounds and maximum utilization of Deferrable Server etc. Section IV provides conclusion.

# General idea of deferrable server

## Definition

It is a simplest bandwidth-preserving server that improves response time of aperiodic jobs as compared to polling server.

## Bandwidth-preserving algorithms

A bandwidth-preserving algorithms provide a mechanism for preserving the resource bandwidth (server capacity) allocated for aperiodic service. Similar in spirit but differ in the way, their server capacity is replenished or preserved. The bandwidth-preserving approaches provide improved response times for aperiodic tasks. One of many examples of bandwidth-preserving algorithms is Deferrable Server.

Before going deeper about what does Deferrable Server do to schedule aperiodic tasks with periodic tasks , it is better to really know what is exactly a periodic task and an aperiodic task. Tasks is basically a job run by a system.

## Periodic Tasks & Aperiodic Tasks

### Periodic tasks

In many real-time control applications, periodic activities represent the major computational demand in the system. Periodic tasks typically arise from sensory data acquisition, low-level servoing, control loops, action planning, and system monitoring. Such activities need to be cyclically executed at specific rates, which can be derived from the application requirements. When a control application consists of several concurrent periodic tasks with individual timing constraints, the operating system has to guarantee that each periodic instance is regularly activated at its proper rate and is completed within its deadline (which, in general, could be different than its period).

### Aperiodic tasks or Non-periodic task

Aperiodic real-time tasks are real-time tasks that occur at any arbitrary moment. The time gap between two aperiodic real-time jobs might be 0. Aperiodic real-time jobs are often soft real-time activities. It's also feasible that these jobs happen often or that the delay between two aperiodic real-time tasks is large.

# Fixed priority Servers

As discussed earlier, there are 2 types of tasks which are periodic and aperiodic. For periodic task scheduling, there are dynamic and static scheduling. Priority Servers are to ensure that all hard jobs can be scheduled in the worst-case scenario and to give excellent average response times for soft and non-real-time operations. Creating a periodic server to process aperiodic tasks. In general, the server is scheduled using the same mechanism that is used for periodic jobs, and once operational, it fulfills aperiodic requests within its budget limit depends on deadline, arrival time or computation time. These are examples of fixed priority servers:

## Polling Server(PS)

To handle aperiodic jobs by using polling server is very common. When executed, it checks the aperiodic jobs queue. If an aperiodic task in a queue, it will be executed when the capacity server is available. When the aperiodic queue is empty during polling process, the server suspends and gives up its’ budget or capacity, means that the budget is replenished.

## Deferrable Server(DS)

We’ll discuss more in the next section.

## Priority Exchange(PE)

Priority Exchange Server keeps the budget until the end of server period like Deferrable Server. Unlike Deferrable Server, the priority slips over time. When not used, the priority is exchanged for of the executing periodic task.

## Sporadic Server(SS)

For Sporadic Server, if a server is in the running or ready queue, it is active; otherwise, it is idle. The server becomes active when an aperiodic job occurs and the budget is not zero.

## Slack Stealing

Slack Stealing algorithm does not create a periodic server for aperiodic task service. Rather it creates a passive task, referred to as the Slack Stealer, which attempts to make time for servicing aperiodic tasks by “stealing” all the processing time it can from the periodic tasks without causing their deadlines to be missed.

Identify applicable funding agency here. If none, delete this text box.

# Model of deferrable server

The method used is to create a separate periodic server job for handling aperiodic tasks. It's the same as a polling job if the server task's capability is only accessible at periodic intervals (hereafter called a Polling Server). By contrast, the Deferrable Server or DS capacity is available for processing aperiodic tasks arriving at any time in its period, a modification which leads to better aperiodic task response times.

Like any periodic task, a server is characterized by a period Ts and a computation time Cs, called server capacity, or server budget. The priority to the server is assigned according to the rate-monotonic scheduling algorithm. In general, period of the server is chosen in a way that it becomes the highest priority task. The Deferrable Server (DS) maintains its aperiodic execution time for the duration of the server’s period. Thus, aperiodic requests can be serviced at the server’s high priority at any time as long as the server’s execution time Cds for the current period has not been exhausted. If the server’s execution time is not used by the end of its period, the server’s execution time is discarded and lost completely. The server’s high priority execution time Cds is replenished to its full capacity, at the beginning of its period.

## Assumptions

### Set of tasks are pre-emptible.

### Independent

### Feasible schedule for periodic tasks

### Minimize response time for aperiodic tasks

## Rule States the Behavior of Deferrable Server

There are two types of rules when aperiodic task is allowed to be scheduled:

### Consumption Rule

Consumption Rule describes when and how much of the budget is decreased when server is executing. Means that, how much the budget is consumed at the time. The budget is consumed at the rate of one per time unit when server executes. When aperiodic jobs are executing, the budget or capacity is consumed at the same rate as the aperiodic job was executed at.

### Replenishment Rule

Replenishment rule describes when and how much the budget is increased. The budget is set to es at the beginning of a new period Ps. The budget is returned to the server every server period.

## Example of Deferrable Server

### Example of a Deferrable Server scheduled by Rate Monotonic

A picture containing chart

Description automatically generatedA picture containing diagram

Description automatically generated

Figure above illustrates an example of aperiodic service obtained through the Deferrable Server scheduled by Rate Monotonic (RM) scheduling. The aperiodic requests are reported on the third row, whereas the fourth row shows the server capacity as a function of time. Number beside the arrows indicate the computation time associated with the requests.  
  
In example figure above the Deferrable Server has a period Ts= 5 and a capacity server Cs = 2, so it runs with an intermediate priority with respect to the other periodic task. When time t=0, the processor is assigned to task T1, which is the highest-priority task according to Rate Monotonic. At time t=1, T1 completes its execution and there is no pending aperiodic task, hence the capacity server is preserved for future aperiodic tasks arrival. Then, the processor is assigned to task T2 until t=2 and after that, the processor is assigned to Deferrable Server . At this time, aperiodic task requests to be handled. Since the capacity server is active and no periodic task is executed, the aperiodic task can be executed immediately until the capacity server is exhausted at time t=4.

When the capacity of Deferrable is already fully consumed, no other requests can be serviced before the next period. However, for every period server Ts, the server capacity Cs is replenished at its full value and preserved until the next aperiodic arrival.

### Example of high priority Deferrable Server

As described in figure at section 1, how Deferrable Server is scheduled by Rate Monotonic (RM) scheduling. It is proved that Deferrable Server violates or clearly breaches the fundamental principle which is for any schedulability examination of the Rate-Monotonic algorithm has been based on the implicit assumption that a periodic job cannot suspend itself and must run whenever the highest-priority task is available. Indeed, the schedule shown in figure above reveals that Deferrable Server does not run at time t = 0, despite being the highest-priority job available to start but it delays execution until time t = 5, which is the arrival time of the first aperiodic request.

The problem occurs when periodic task delays its execution time even though it could run immediately. A lower priority task could miss its deadline even if the task set was schedulable. Figure below explains how this problem is happened. It shows comparison when the execution of a periodic task to the one of a Deferrable Server with the same period and execution time.

A picture containing timeline

Description automatically generated

Figure (a) illustrates two periodic tasks T1 and T2 are scheduled by Rate Monotonic (RM). Both tasks have computation time C1= C2= 2 and different periods (T1 = 4, T2 =5). If T1 is replaced by Deferrable Server having the same period and execution time, T2 that have low priority task can miss its deadline depending on the sequence of aperiodic arrivals as shown in Figure (b). T2 misses its deadline when t=15 and this problem occurs because at time t = 8, DS does not execute (as a regular periodic job would) but instead saves its capacity for future requests. Task 2 is prevented from running during this interval by this postponed execution, which is followed by the servicing of two successive aperiodic requests in the interval [10, 14].

Because of the Deferrable Server's intrusive behavior, the periodic task set has a reduced schedulability constraint. In the next section, the least upper bound of the process or utilization factor is calculated in the presence of Deferrable Server.

## Calculation of for RM+DS and

The least upper bounds offer necessary conditions for task set schedulability in the sense that all periodic task deadlines will be met if task set utilization is below the bound. To make computing the bound for n tasks easier, we first find the worst-case relationships between the tasks, and then deduce the lower limit against the worst-case connections.

Chart, box and whisker chart

Description automatically generated

As depicted in figure above, it describes the most common case for this derivation which Deferrable Server may execute three times within the period of the highest priority periodic task. The reason is when DS delays its service at the end of the period and executes at the beginning of the next period.

The expressions below describes how utilization of the least

upper bound, can be calculated. As we know, Capacity

Server, can be derived as:

Therefore, we can get utilization which is

Besides, we know that resource equations are

And noting that

Therefore, utilization factor is

Based on approached used for Rate Monotonic, is

minimized over where .Therefore,

Since defining , is minimum when

that is when all have the same value:

By substituting this value in , so

Thus,

Finally, utilization of the least upper bound is given below:

Using n → ∞ as the limit, we discover that the equation

below gives the worst-case bound as a function of

Figure 5.11 shows a visualization of Equation (5.12) as a function of . The RM bound is also shown in the plot for comparison. The existence of DS weakens the RM bound for .4, but it improves the RM bound for > 0.4

**Chart, line chart

Description automatically generated**

We may calculate the absolute lowest value of by

applying Equation (5.12) to :

The value of that reduces the above equation to its simplest

form is

Hence, the minimum value of is

In conclusion, given a set of n periodic tasks with utilization

Up and a Deferrable Server with utilization Us, the periodic

task set's schedulability is guaranteed under RM if

Where

Besides that, the guarantee test for a task set in the presence of a Deferrable Server may be conducted as follows using the Hyperbolic Bound:

Therefore, we can also get maximum utilization. Since we have:

Then, we also have

Thus,

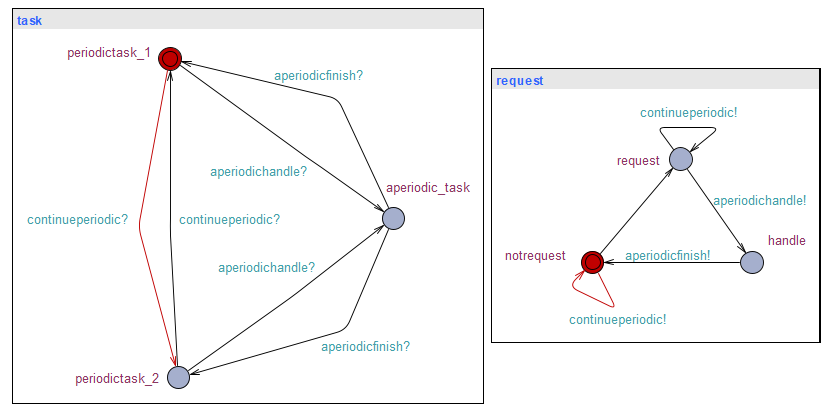
can then be adjusted to the shortest time T1, ensuring that Deferrable Server gets the highest priority from Rate Monotonic (assuming that priority ties are broken in favor of the server), and .

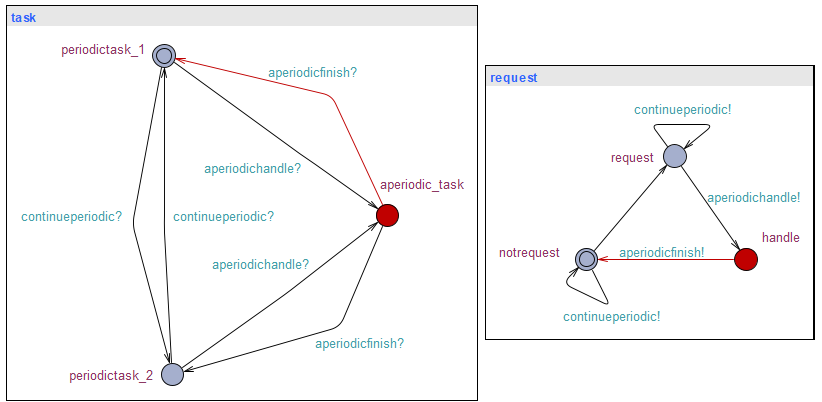
In the next section, UPPAAL implementation is described to make clear how Deferrable Server works in scheduling process in real-time system.

## UPPAAL implementation

As shown in figure below, it describes how Deferrable Server is handling aperiodic task under UPPAAL environment. The first figure describes when periodic task 1 and task 2 run. When there is a request from aperiodic task as depicted in the second figure, periodic task 1 continue to run until it finishes its task. After that, processor will assign to aperiodic job to be handled like in the third figure. Once it finishes, it will go back to run periodic task.

Chart, diagram, radar chart

Description automatically generated



## Comparison with other fixed priority server

Table

Description automatically generated

As shown in figure below, we can see the comparison of fixed priority servers. There are many factors can be considered to evaluate which server is the most appropriate to handle soft aperiodic tasks in a hard real-time environment.

## 

# CONCLUSION

This paper has presented the theoretical foundations for the Deferrable Server algorithm which provides a solution to the problem of jointly scheduling hard deadline periodic tasks and hard and soft deadline aperiodic tasks. To provide a fair comparison, both necessary and sufficient conditions and least upper bounds were developed for the Deferrable Server algorithm and the more conventional polling technique of scheduling aperiodic service in hard-time environments. Taking advantage of the fact that there is typically no advantage for the system for periodic tasks completing early, the DS algorithm converts the excess periodic task slack time into highly responsive aperiodic class performance.

The algorithm has been used to introduce highly responsive, guaranteed alert task aperiodic service while still maintaining periodic task guarantees as well as providing response time improvements of an order of magnitude for soft deadline aperiodic tasks. The algorithm has been shown to provide nearly optimal aperiodic response time performance for relatively short aperiodic mean service times up to very high server traffic intensities. As the mean service times were increased, the DS response time performance diverged more quickly from the optimal noninterfering case. Other application simulation studies in both processor and LAN media access scheduling have shown similar results and sensitivities.

##### Acknowledgement

##### References

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*