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 to scheduling is required, so that enable to meet the time requirements of periodic task. The scheduling algorithm is created and applied either to periodic tasks or aperiodic tasks but not for both. Deferrable Server algorithm is an algorithm for aperiodic tasks that can improve aperiodic time response performance during scheduling process.

# 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.

## What is 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. The examples of bandwidth-preserving algorithms are :

* Deferrable Server
* Priority Exchange Algorithm
* Extended Priority Exchange Algorithm
* Sporadic Server

# periodic & aperiodic tasks

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

## 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

Typically aperiodic tasks are event-driven, soft or hard RT

# FIXED PRIORITY SERVERS

The scheduling algorithms treateD deal with homogeneous sets of tasks, where all computational activities are either aperiodic or periodic. Many real-time control applications, however, require both types of processes, which may also differ for their criticality. Typically, periodic tasks are time-driven and execute critical control activities with hard timing constraints aimed at guaranteeing regular activation rates. Aperiodic tasks are usually event-driven and may have hard, soft, or non-real-time requirements depending on the specific application.

When dealing with hybrid task sets, the main objective of the kernel is to guarantee the schedulability of all critical tasks in worst-case conditions and provide good average response times for soft and non-real-time activities. Off-line guarantee of event-driven aperiodic tasks with critical timing constraints can be done only by making proper assumptions on the environment; that is, by assuming a maximum arrival rate for each critical event. This implies that aperiodic tasks associated with critical events are characterized by a minimum interarrival time between consecutive instances, which bounds the aperiodic load. Aperiodic tasks characterized by a minimum interarrival time are called sporadic. They are guaranteed under peak-load situations by assuming their maximum arrival rate.

If the maximum arrival rate of some event cannot be bounded a priori, the associated aperiodic task cannot be guaranteed off-line, although an online guarantee of individual aperiodic requests can still be done. Aperiodic tasks requiring online guarantee on individual instances are called firm. Whenever a firm aperiodic request enters the system, an acceptance test can be executed by the kernel to verify whether the request can be served within its deadline. If such a guarantee cannot be done, the request is rejected.

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

# What does deferrable server do

Deferrable Server creates a periodic server task of Capacity Cds and Period Tds. 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. The Deferrable Server is also invoked with a fixed period. It differs from the Periodic Server in that if no tasks are ready to use the server then it may suspend its execution, preserving its capacity. The Deferrable Server’s capacity may be preserved throughout its period. If an application task becomes ready late in the server’s period it can be executed until either the server’s capacity is exhausted or the end of the server’s period is reached. At the end of the server’s period any remaining server capacity is discarded and the server’s capacity is then replenished. Again execution of the server may be delayed and or pre-empted by the execution of other servers of a higher priority. Schedulability analysis of the Deferrable Server needs to take account of the well-known phenomenon of back-to-back hits. By preserving its capacity until near the end of its period a high priority Deferrable Server can cause back-to-back interference of 2CS on lower priority servers. Effectively a Deferrable Server has a jitter equal to −CT SS

# schedulability analysis

Any schedulability analysis related to the Rate-Monotonic algorithm has been done on the implicit assumption that a periodic task cannot suspend itself, but must execute whenever it is the highest-priority task ready to run (assumption A5 in Section 4.1). It is easy to see that the Deferrable Server violates this basic assumption. In fact, the schedule illustrated in Figure below shows that DS does not execute at time t = 0, although it is the highest-priority task ready to run, but it defers its execution until time t = 5, which is the arrival time of the first aperiodic request. If a periodic task defers its execution when it could execute immediately, then a lowerpriority task could miss its deadline even if the task set was schedulable.

Figure below illustrates this phenomenon by comparing the execution of a periodic task to the one of a Deferrable Server with the same period and execution time. The periodic task set considered in this example consists of two tasks, τ 1 and τ2, having the same computation time (C1 = C2 = 2) and different periods (T1 = 4, T2 = 5). As shown in Figure below, the two tasks are schedulable by RM. However, if τ1 is replaced with a Deferrable Server having the same period and execution time, the low-priority task τ2 can miss its deadline depending on the sequence of aperiodic arrivals.

Figure below shows a particular sequence of aperiodic requests that cause τ2 to miss its deadline at time t = 15. This happens because, at time t = 8, DS does not execute (as a normal periodic task would do) but preserves its capacity for future requests. This deferred execution, followed by the servicing of two consecutive aperiodic requests in the interval [10, 14], prevents task τ2 from executing during this interval, causing its deadline to be missed. Such an invasive behavior of the Deferrable Server results in a lower schedulability bound for the periodic task set. The calculation of the least upper bound of the processor utilization factor in the presence of Deferrable Server is shown in the next section.

A picture containing chart

Description automatically generated

Figure : Example deferrable server scheduled by RM

A picture containing diagram

Description automatically generatedFigure : Example of high-priority Deferrable Server.

A picture containing timeline

Description automatically generated

Figure : DS is not equivalent to a periodic task. In fact, the periodic set {τ1, τ2} is schedulable by RM (a); however, if we replace τ1 with DS, τ2 misses its deadline (b).

# dimensioning a deferrable server

Following the same procedure described in Section 5.3.2, the maximum utilization Umax s for a Deferrable Server can easily be computed from Equation (5.14), which can be written, defining P as in Equation (5.7), as: P ≤ Us + 2 2Us + 1; that is, Us ≤ 2 − P 2P − 1 . Hence, Umax s = 2 − P 2P − 1 . (5.15) Then, Ts can be set equal to the smallest period T1, so that DS is executed by RM with the highest priority (assuming that priority ties are broken in favor of the server), and finally Cs = UsTs.

# aperiodic guarantee

The online guarantee of a firm aperiodic job can be performed by estimating its worstcase response time in the case of a DS with the highest priority. Since DS preserves its execution time, let cs(t) be the value of its capacity at time t, and let Ja an aperiodic job with computation time Ca and relative deadline Da, arriving at time t = ra, when no other aperiodic requests are pending. Then, if next(r a) = &ra/Ts'Ts is the next server activation after time ra, the two cases illustrated in Figure 5.12 can occur: 1. Case (a): cs(t) ≤ next(ra) − ra. In this case, the capacity is completely discharged within the current period and a portion C0 = cs(t) of Ja is executed in the current server period. 2. Case (b): cs(t) > next(ra) − ra. In this case, the period ends before the server capacity is completely discharged; thus a portion C0 = next(ra) − ra of Ja is executed in the current server period. In general, the portion C0 executed in the current server period is equal to C0 = min{cs(t), next(ra) − ra}

Diagram

Description automatically generated

Figure : Execution of Ja in the first server period.

A picture containing graphical user interface

Description automatically generated

Figure : Response time of an aperiodic job scheduled by a Deferrable Server with the highest priority.

Using the same notation introduced for Polling Server, we define:    ∆a = next(ra) − ra Fa = 1 Ca−C0 Cs 2 − 1 δa = Ca − C0 − FaCs. Hence, as depicted in Figure 5.13, the response time Ra of job Ja can be computed as Ra = ∆a + FaTs + δa, which can be also written as: Ra = ∆a + Ca − C0 + Fa(Ts − Cs). (5.16) Note that the term Fa(Ts − Cs) in Equation (5.16) represents the delay introduced by the Fa inactive server intervals, each of size (Ts − Cs). Then, the schedulability of the aperiodic job can be guaranteed if and only if R a ≤ Da.

# 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.

##### 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)*