Deferrable Server

Muhammad Farid Izwan Bin Mohamad Shabri
Hochschule Hamm-Lippstadt
B.Eng. Electronic Engineering
Lippstadt, Germany
muhammad-farid-izwan.bin-mohamad-shabri@stud.hshl.de

***Abstract*—This paper introduces Deferrable Server algorithm which is to improve aperiodic task response time in real-time system. It assigns higher priority aperiodic task to be completed by delaying the completion time of periodic task, so that the deadline for aperiodic tasks is 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.**

I. Introduction

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. To schedule the periodic and aperiodic activities together is very challenging and tough so that all 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 aperiodic activities is by background scheduling, that is, when there are no periodic instances available to execute. The main disadvantage of this strategy is that the response of aperiodic requests might take too much time for some applications if large periodic loads are scheduled. As a result, it is suitable to schedule in background only when aperiodic tasks do not have strict time requirements and the periodic load is low. Besides, the background scheduling is very simple which not very hard to implement. To implement the background scheduling system, two queues are required: one for periodic tasks which is for a higher priority and one for low priority 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 from the aperiodic queue can be executed. When a new periodic instance is activated, all aperiodic tasks are instantly preempted. Moreover, the improvement of average response time of aperiodic tasks can be done in terms of background scheduling by using a server. The server objective is to service aperiodic requests as quickly as feasible while periodic task can meet the deadline [1].

This paper describes Deferrable Server algorithm that can improve aperiodic time response performance during scheduling process. Lehoczky, Sha, and Strosnider proposed the Deferrable Server (DS) method to reduce the average response time of aperiodic queries as compared to polling services [3]. 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 server or also called budget if no requests are queued when the server is started. As long as the capacity has not been depleted, the service aperiodic requests can be done at the same server's priority. The server budget is renewed at full value at the start of each server term, so that the next aperiodic tasks can be executed when there is a request [3].

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.

II. Overview of Deferrable Server

In this section we will explain the overview of Deferrable Server. The definition is explained and tasks are described ,so that we get better understanding about Deferrable Server.

A. Definition

Deferrable Server is a simplest bandwidth-preserving server that improves response time of aperiodic jobs . A bandwidth-preserving algorithms provide a mechanism for preserving the resource bandwidth which is server capacity or budget allocated for aperiodic service during scheduling process. The main point is the server capacity is replenished or preserved. The bandwidth-preserving methods provide improved response times performance for aperiodic tasks when executed. One of many examples of bandwidth-preserving algorithms is Deferrable Server. [1]

Before going deeper about what does Deferrable Server do to schedule aperiodic tasks with periodic tasks in scheduling process without causing any time problem , it is better to really know what is exactly a periodic task and an aperiodic task. In general, tasks is basically a job run by a system. Task is divided into two which are periodic task and aperiodic task.

B. Task

In general, tasks is basically a job run by a system. Task is divided into two which are periodic task and aperiodic task.

1) Periodic Task: Periodic task represents the major computational demand in the system for most real-time control applications. It consists of an infinite sequence of identical activities, called instances or jobs. Periodic task is run at constant rate or at specific time and must meet its deadline [2].

2) Aperiodic Task 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. As mentioned above, aperiodic task time response can be improved by the server. The Deferrable Server is an example of Fixed Priority Server [2].

In the next section, we will explain fixed priority server and its examples.

III. Fixed Priority Server

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 time in running soft and non-real-time operations by 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 [3]. These are examples of fixed priority servers:

A. Polling Server

To handle aperiodic jobs by using polling server is very common. It checks the aperiodic jobs queue when during tasks' execution. If the queue has an aperiodic activity, it will be executed when the capacity server is available. In polling process, if the is no aperiodic task in the queue, the server suspends and gives up its' budget or capacity, means that the budget is replenished [1].

B. Deferrable Server

We'll discuss more details in the next section.

C. Priority Exchange(PE)

Priority Exchange Server maintains the capacity server in the period server like Deferrable Server. But over time, capacity deteriorates unlike Deferrable Server. The priority is exchanged for of the executing periodic task when it is not utilized [1].

D. Sporadic Server(SS)

For Sporadic Server, the server becomes active when an aperiodic job occurs, and the budget is not zero. If the server is in the running or ready queue, it is active. Otherwise, it is idle [1].

E. Slack Stealing

Unlike others fixed priority server, a periodic server is not created in Slack Stealing algorithm to schedule aperiodic task. However, it creates a passive task, referred to as the Slack Stealer. Slack Stealer tries to free up time for aperiodic tasks by "stealing" as much processing time as it can from periodic tasks without causing them to miss their deadlines [1].

In the next section, Deferrable Server is explained more details.

IV. Model of Deferrable Server

As discussed earlier, Deferrable Server is a bandwidth-preserving algorithm. The method used is to produce a separate periodic server job to service aperiodic tasks which can satisfy the time requirements. 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 [3].

In addition, a server, like any other periodic task, has a period T_s and a computation time C_s 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) keeps 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 if the server's execution time C_s 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,C_s is replenished to its full capacity, at the beginning of its period Ts [1].

A. Assumptions

- Set of tasks are pre-emptible.
- Independent
- Feasible schedule for periodic tasks
- Minimize response time for aperiodic tasks

B. Rule States the Behavior of Deferrable Server

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

1) 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. During execution process, the server budget or server capacity C_s is consumed at the rate of one per time unit. When aperiodic jobs are running, the budget or capacity is utilized at the same rate as the aperiodic job was executed at [4].

2) Replenishment Rule: Replenishment rule describes when and how much the budget is increased. The budget is renew at the beginning of a new period T_s. The capacity server or server budget C_s is returned to the server every server period, so that aperiodic task can be handled [4].

C. Example of Deferrable Server

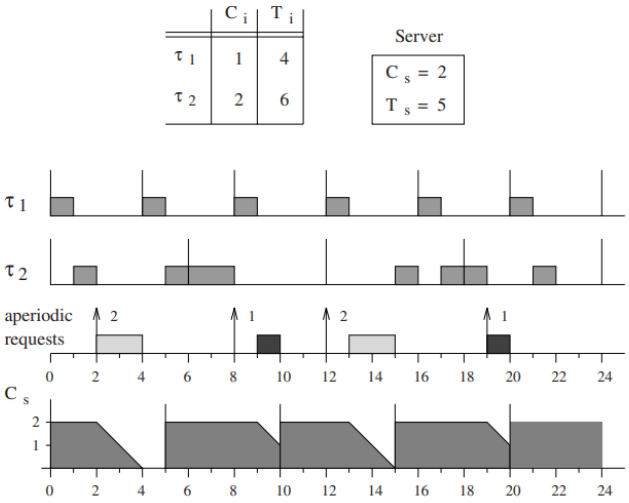

Fig. 1. Example of a Deferrable Server scheduled by Rate Monotonic. [1]

1) Example of Deferrable Server scheduled by Rate Monotonic: As depicted in "Fig. 1", it illustrates an example of aperiodic jobs service by the Deferrable Server that scheduled by Rate Monotonic (RM) scheduling. On the third row, it describes the requested aperiodic tasks, 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 the "Fig. 1", the Deferrable Server has a period T_s= 5 and a capacity server C_s = 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 τ_1, which is the highest-priority task according to Rate Monotonic. At time t=1, $\tau_1$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 τ_2 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 T_s, the server capacity C_s is replenished at its full value and preserved until the next aperiodic arrival [1].

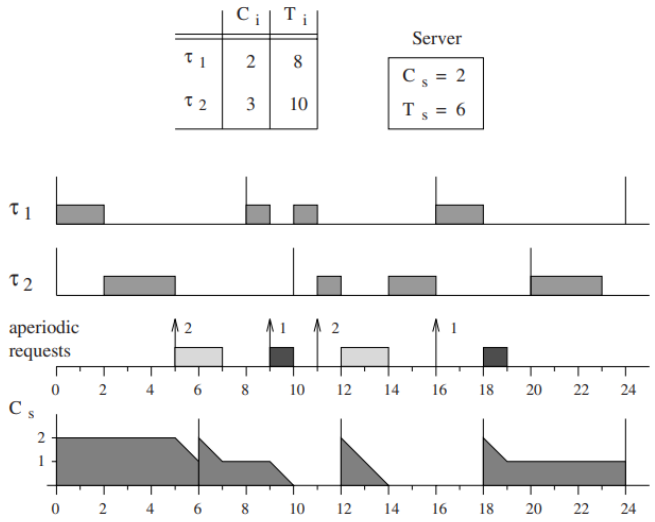

Fig. 2. Example of high-priority Deferrable Server. [1]

2) Example of high priority Deferrable Server: As described in "Fig. 2", it explains 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 it is obligatory to run whenever the highest-priority task is available.

Indeed, the schedule shown in "Fig. 2" 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 [1].

3) Example of Deferrable Server when defer happens: The problem occurs when periodic task delays its execution time even though it could run instantly. The deadline for lower priority task could not satisfy even if the task set was schedulable. "Fig . 3"explains how this problem could happen. It shows comparison when the running periodic task to the one of a Deferrable Server with the exact identical period and execution time.

As depicted in "Fig . 3(a)" , two periodic tasks τ_1 and τ_2 are scheduled by Rate Monotonic (RM). Both tasks have computation time C_1= C_2= 2 and different periods (τ_1 = 4, τ_2 =5). If τ_1 is replaced by Deferrable Server having the same period and execution time, the low priority task, τ_2 can miss its deadline depending on the order of aperiodic task when the tasks are arrived as shown in "Figure . 3(b)". τ_2 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 running two successive

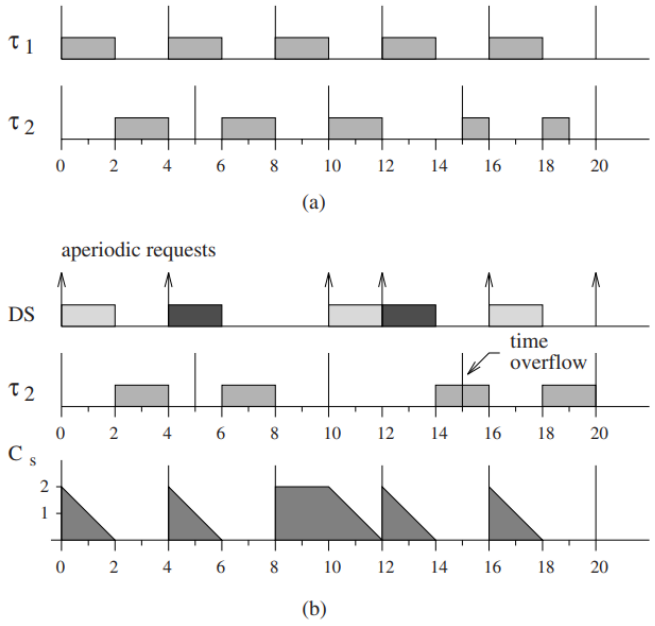

Fig. 3. Periodic task is scheduled by Rate Monotonic (a); however, f we replace τ_1 with DS, τ_2 misses its deadline. [1]

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 and maximum utilization is calculated in the presence of Deferrable Server. This is to check whether the deadline will be met or not [1].

D. Calculation of U_{lub} for RM+DS and U_{lub}^{max}

The least upper bounds offer necessary requirements 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 [1].

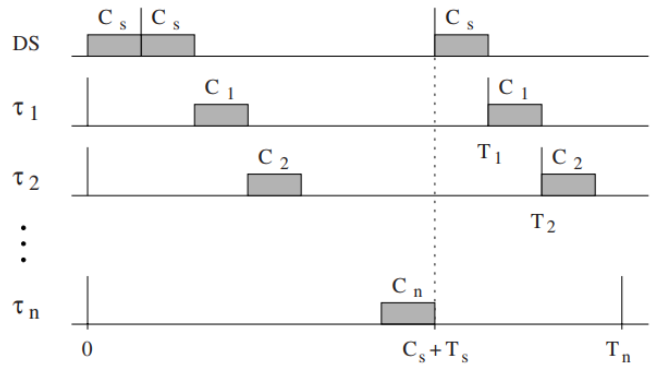

Fig. 4. Worst-case task relations for a Deferrable Server. [1]

As depicted in "Fig. 4" 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 Deferrable Server delays its service at the end of the period and executes at the beginning of the next period.

The expressions below describe how utilization of the least upper bound, U_{lub} can be calculated. As we know, Capacity Server, C_s can be derived as:

$$C_s = T_1 - \big(T_s + C_s = \tfrac{T_1 - T_s}{2}$$

$$\begin{cases} C_s = T_1 - T_2 \\ \dots \\ C_{n-1} = T_n - T_{n-1} \end{cases}$$

$$C_n = T_s - C_s - \sum i = 1^{n-1} C_i = \tfrac{3T_s + T_1 - 2T_n}{2}$$

Therefore, we can get utilization which is

$$U = \frac{C_s}{T_s} + \frac{C_1}{T_1} + \dots + \frac{C_n}{T_n} =$$

$$= U_s + \frac{T_2 - T_1}{T_1} + \dots + \frac{T_n - T_{n-1}}{T_{n-1}} + \frac{3T_s - T_1 - 2T_n}{2T_n} =$$

$$= U_s + \frac{T_2}{T_1} + \dots + \Big(\frac{3T_s}{2T_1} + \frac{1}{2}\Big)\frac{T_1}{T_n} - n$$

Besides, we know that resource equations are

$$\begin{cases} R_s & = \frac{T_1}{T_s} \\ R_i & = \frac{T_{i+1}}{T_i} \\ K & = \frac{1}{2}\big(3T_s/T_1 + 1\big) \end{cases}$$

and noting that

$$R_1 R_2 \dots R_{n-1} = \frac{T_n}{T_1}$$

Therefore, utilization factor is

$$U = U_s + \sum_{i=1}^{n-1} R_i + \frac{K}{R_1 R_2 \dots R_{n-1}} - n$$

Based on approached used for Rate Monotonic, U is minimized over R_i where $i = 1, \dots, n-1$.Therefore,

$$\frac{\partial U}{\partial R_i} = 1 - \frac{K}{R_i^2\big(\prod_{j \neq 1}^{n-1} R_j\big)}$$

Since defining $P = R_1 R_2 \dots R_{n-1}$, U is minimum when

$$\begin{cases} R_1 P & = K \\ R_2 P & = K \\ \dots & \\ R_{n-1} P & = K \end{cases}$$

that is when all R_i have the same value:

$$R_1 = R_2 = \dots = R_{n-1} = K^{\frac{1}{n}}$$

By substituting this value in U, so

$$U_{lub} - U_s = (n-1)K^{1/n} + \frac{K}{K^{(1-1/n)}} - n$$

$$U_{lub} = U_s + n\Big(K^{\frac{1}{n}} - 1\Big) \tag{1}$$

Thus,

$$K = \frac{U_s + 2}{2U_s + 1}$$

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

$$U_{lub} = U_s + n\left[\left(\frac{U_s + 2}{2U_s + 1}\right)^{\frac{1}{n}} - 1\right] \tag{2}$$

Using $n \to \infty$ as the limit, we discover that the equation below gives the worst-case bound as a function of U_s

$$\lim_{n\to\infty} U_{lub} = U_s + \ln\left(\frac{U_s + 2}{2U_s + 1}\right) \tag{3}$$

"Fig . 5" shows a visualization of Eq. (3) as a function of U_s. The RM bound is also shown in the plot for comparison. The existence of DS weakens the RM bound for $U_s < 0.4$, but it improves the RM bound for $U_s > 0.4$.

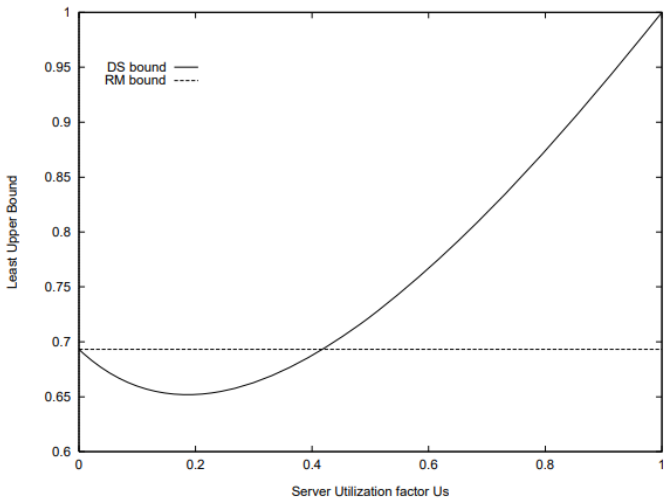

Fig. 5. Schedulability bound for periodic tasks and DS as a function of the server utilization factor U_s. [1]

We may calculate the absolute lowest value of U_{lub} by applying Eq. (3) to U_s:

$$\frac{\partial U_{lub}}{\partial U_s} = 1 + \frac{(2U_s + 1)}{(U_s + 2)} \frac{(2U_s + 1) - 2(U_s + 2)}{(2U_s + 1)^2}$$
$$= \frac{2U_s^2 + 5U_s - 1}{(U_s + 2)(2U_s + 1)}$$

The value of U_s that reduces the above equation to its simplest form is

$$U_s^* = \frac{\sqrt{33} - 5}{4} \simeq 0.186$$

Hence, the minimum value of U_{lub} is $U_{lub}^* \simeq 0.652$ In conclusion, given a set of n periodic tasks with utilization Up and a Deferrable Server with utilization U_s, the periodic task set's schedulability is guaranteed under RM if

$$U_p \leq n(K^{\frac{1}{n}} - 1) \tag{4}$$

Where

$$K \leq \frac{U_s + 2}{2U_s + 1}$$

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:

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{U_s + 2}{2U_s + 1} \tag{5}$$

Therefore, we can also get maximum utilization U_s^{max} by deriving from Eq. (5). Since we have:

$$P \leq \frac{U_s + 2}{2U_s + 1}$$

Then, we also have

$$U_s \leq \frac{2 - P}{2P - 1}$$

Thus,

$$U_s^{max} = \frac{2 - P}{2P - 1} \tag{6}$$

T_s can then be adjusted to the shortest time τ_1, ensuring that Deferrable Server gets the highest priority from Rate Monotonic (assuming that priority ties are broken in favor of the server), and $C_s = U_s T_s$.

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

E. UPPAAL Model of Deferrable Server

In this section, the UPPAAL model of Deferrable Server will be explained. As shown in "Fig . 6" describes when periodic task 1 and task 2 run. When there is a request from aperiodic task as depicted in the "Fig . 7", 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 "Fig . 8". Once it finishes, it will go back to run periodic task.

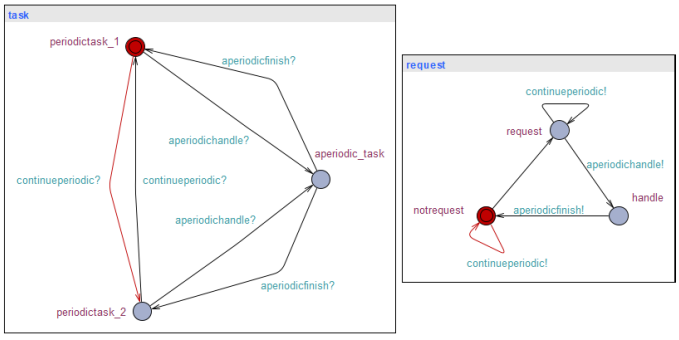

Fig. 6. When the periodic task is handled.

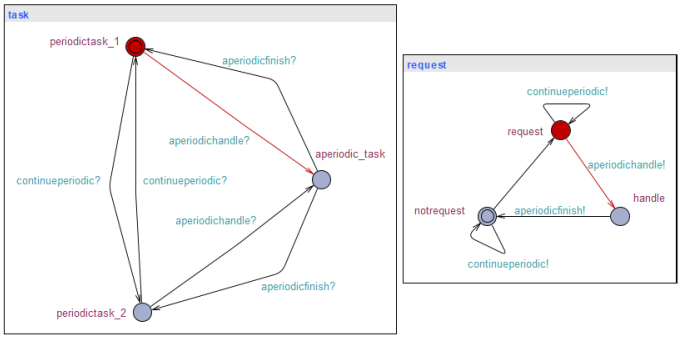

Fig. 7. The aperiodic task request to run.

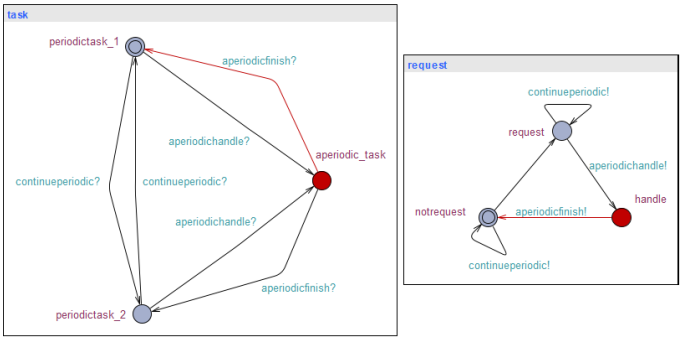

Fig. 8. The aperiodic task run after requesting to the server.

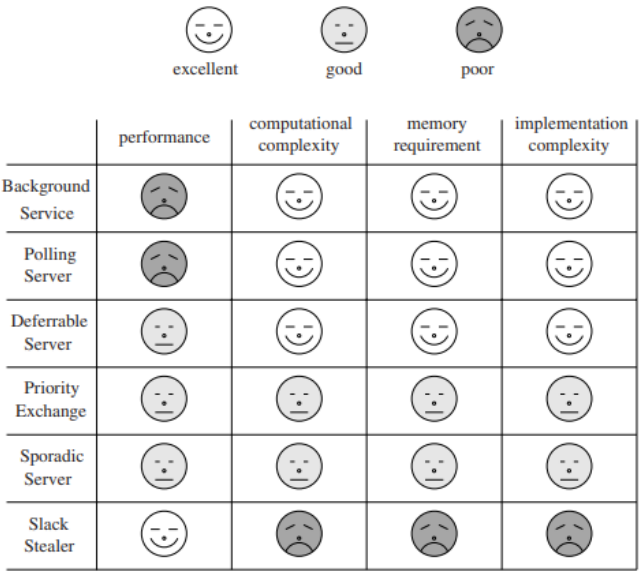

Fig. 9. Comparison of fixed-priority servers. [1]

F. Comparison with other fixed priority server

As shown in "Fig . 9", 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.

For Deferrable Server, it provides excellent result in terms of computational complexity, memory requirement and implementation complexity. Besides, the performance of Defeerable Server is good. By comparing to others, Deferrable Server is quite good that could handle aperiodic task but of course, it depends on what kind of system being handled.

V. Conclusion

This paper has presented the theoretical foundations for the Deferrable Server algorithm which provides a solution to the problem of jointly scheduling periodic tasks and aperiodic tasks. Deferrable Server algorithm provides highly responsive aperiodic class performance. Since we know, there is no benefits if the periodic task system completes its task earlier, as long as its deadline can be met.Lastly, Moreover, Deferrable Server secure warning task aperiodic services while still maintaining periodic task guarantees as well as providing an increase in order response times for aperiodic tasks. Last but not least, it has proven that Deferrable Server provides almost optimal aperiodic response time performance for relatively short aperiodic mean service times.

Acknowledgment

I would love to express my special thanks to Prof. Dr. Stefan Henkler for the guidance and support that has given in making this paper.

References

[1] G. Buttazzo, Hard real-time computing systems. [S.l.]: Springer, 2013.
[2] H. Kopetz, Real-Time Systems. Boston: Springer, 2011.
[3] J. K. Strosnider, J. P. Lehoczky and Lui Sha, "The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments," in IEEE Transactions on Computers, vol. 44, no. 1, pp. 73-91, Jan. 1995, doi: 10.1109/12.368008..
[4] H. Zhu, S. Goddard and M. B. Dwyer, "Response Time Analysis of Hierarchical Scheduling: The Synchronized Deferrable Servers Approach," 2011 IEEE 32nd Real-Time Systems Symposium, 2011, pp. 239-248, doi: 10.1109/RTSS.2011.29..