Improved Priority Exchange Server

Abdul-Azeez Olanlokun
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
abdul-azeez.olanlokun@stud.hshl.de

***Abstract*—This paper is focused on the scheduling algorithm of the improved priority exchange server. In critical systems, it is important to meet deadlines, as there are a lot of processes running at the same time, these processes need to be given priority in other to avoid system overloads, which could lead to catastrophic occurrences in the system. In other to avoid these overloads, we need to schedule the system, giving priorities to tasks. In other to achieve an efficient scheduling system, we have extensively analyzed the use of the improved priority exchange server algorithm, which is a modification of the DPE server (Dynamic Priority Exchange Server, explained further in this paper), this is done by using the EDL scheduler (Earliest Deadline Late, explained further in this paper) idle times. This led to the increase in an efficient replenishment policy of this server, and also changed the dynamics of the scheduler which in turn allows regular running of the system at its highest priority.**

***Index Terms*—Improved priority, server, efficiency.**

I. Introduction

A real-time system's objective in the physical world is to have a physical effect within a specified time frame. A real-time system typically consists of a controlling system (computer) and a controlled system (environment). Based on the information available about the environment, the controlling system interacts with it. Sensors will provide readings at repeated intervals on a real-time computer that controls a device or process, and the computer must respond by sending signals to actuators. Unexpected or unusual events may occur, and these must also be addressed. In all situations, the response must be delivered within a specific time frame. The computer's ability to meet these expectations is determined by its ability to do the necessary computations in the allotted time. If a number of events occur at the same time, the computer must arrange the computations such that each response is supplied within the required time limits. Even yet, it is possible that the system will be unable to handle all unanticipated demands. In this scenario, we say the system lacks sufficient resources; a system with infinite resources and the ability to process at infinite speed might satisfy any such temporal limitation. Failure to meet the timing constraint for a response can have a variety of repercussions; there may be no effect, the effects may be minimal or correctable, or the effects may be severe. The aim of this paper is to briefly introduce an improved, efficient algorithm for the joint scheduling of random soft aperiodic requests and hard periodic tasks under the PE (Priority Exchange, In this paper, Improved priority exchange server) server. The proposal is to analyze exclusively, the Improved Priority Exchange Server, and to compare it with two different predecessors' algorithms under the priority server, having different implementation overheads and different performances. The two algorithms, namely the Dynamic Priority Exchange and, an optimal algorithm, the EDL Server, and to compare it with a close approximation of it, the Improved Priority Exchange, which has much less runtime overhead. The EDL and the IPEs are both based on off-line computations of the slack time of the periodic tasks. By balancing efficiency and implementation overhead, the suggested algorithm provides a valuable framework to aid an HRT(Hard Real Time) system designer in picking the best appropriate technique for his or her needs. The paper is organized as follows. In the next section all the assumptions are stated, followed by the brief introduction of the Dynamic Priority Exchange (DPE) algorithm, which is an extension of the Priority Exchange algorithm proposed by Lehozcky et al. [6]. And then we describe briefly EDL (Earliest Deadline Late) server, an optimal algorithm, then to the focused algorithm, IPE server, a nearly optimal algorithm which is derived from DPE and the EDL server, using their insights. Simulation results are discussed in the later sections. Finally, considerations and conclusions.

II. Assumptions and Terminology used

The following assumptions are to be considered in the definition of the algorithms (NB. This is a definition in [1])

- *Every periodic tasks τ_i : i = 1,..,n have hard deadlines;*
- *Every aperiodic tasks J_i : i = 1,....,m have deadlines;*
- *For each periodic task τ has a constant period of T_i and also a constant worst case execution time C_i, which is considered to be known, because it can be derived by the means of static analysis of the source code;*
- Every periodic tasks are activated simultaneously at time t = 0; i.e ., the first instance of each periodic task has a request time $r_i(0) = 0$;
- *The K^{th} periodic instance request time is given by $r_i(k) = r_i(k\text{-}1) + T_i$;*

- *The K^{th} periodic instance deadline is given by $d_i(k) = r_i(k) + T_i$;*
- *The arrival time of each aperiodic task is not known;*
- *Each aperiodic task worst case execution time is considered known at its arrival time.*

All properties of the suggested algorithms will be demonstrated under the preceding assumptions for clarity's sake. They may, however, simply be extended to handle periodic activities with non-null phasing and deadlines that differ from the end of periods. The guarantee tests would only give sufficient criteria for the schedule's practicality in this situation.

III. Early Work

We will briefly describe the early works to this algorithm.

A. The Priority Exchange Algorithms

In this section we will briefly introduce two of the proposed predecessors Priority Exchange algorithms, which are the DPE server and the EDL server, this is to give us basis for our main focus algorithm, which is the IPE server. Lehoczky et al. introduce the Priority Exchange (PE) algorithm in [6], which is a server for aperiodic requests under the RM(Rate Monotonic) algorithm.

1) The Dynamic Priority Exchange Server: The DPE server is a modified version of the PE server that is optimized for the EDF algorithm. If there are no aperiodic requests outstanding, the algorithm's fundamental notion is to let the server trade its run-time with the run-time of lower priority periodic jobs (under EDF, this means a longer deadline). The server's run-time is only traded with periodic tasks in this way, and it is never squandered (unless there are idle times). It's merely saved, although at a lesser priority, and can be retrieved later when aperiodic requests arrive.

An example of the DPE server is shown in the Fig . 1 below

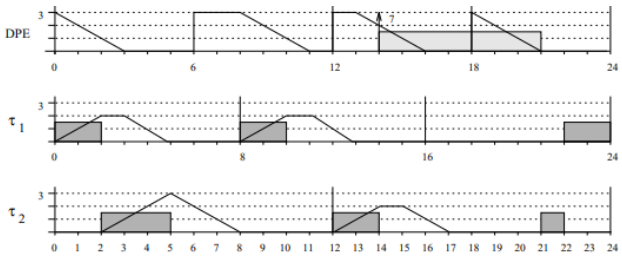


Fig. 1. Dynamic Priority Exchange Server example[1].

2) The Earliest Deadline Late Server: The EDL algorithm's fundamental idea is to take advantage of the total bandwidth algorithm's laxities. To achieve so, the idle times of a specific EDF schedule from a periodic task set are calculated, and an ideal replenishment policy for the capacity of an aperiodic server is developed from these values. The EDL is one of the Implementations of the EDF algorithms, whereby, active tasks are processed as late as possible.EDL has a formal proof of optimality, in the sense that it ensures the maximum idle time in a given interval.

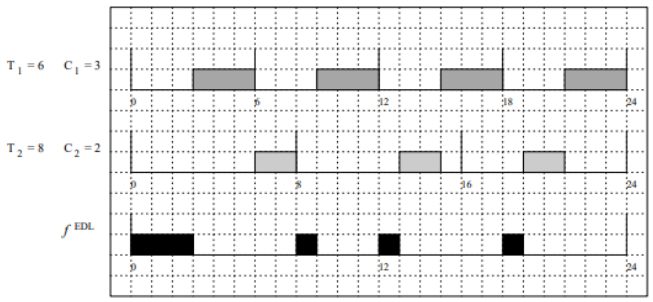


Fig. 2. Availability function under EDL[6].

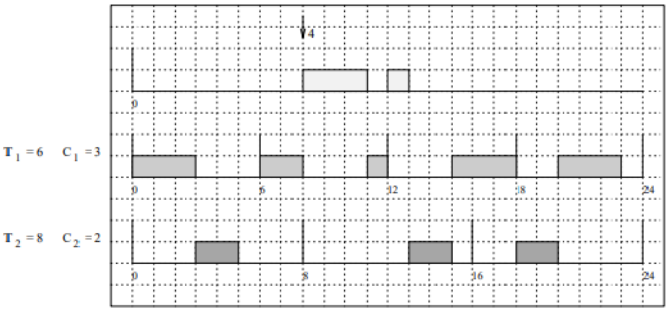


Fig. 3. An Example of schedule produced with an EDL server[6].

IV. The Improved Priority Exchange Server Algorithm

The approach presented in the previous section on EDL is optimal, but it has too much overhead to be deemed practical. However, as shown later in the description of the simulations, its core principle might be advantageously adapted to construct an implementable algorithm while still keeping a nearly optimal behavior. The complexity of determining the idle times at each new aperiodic arrival makes the EDL server impractical. This computation must be performed each time in order to account for periodic instances that are either partially executed or completed at the time of arrival. In addition to the idle time of an ideal EDL scheduler, the time "advanced" to the periodic instances becomes idle time that the server might utilize to plan aperiodic requests. Using the priority exchange mechanism, we can avoid the heavy idle time computation. In fact, using this method, the system may easily keep track of time advanced to periodic tasks and reclaim it at the appropriate priority level. The EDL algorithm idle times can be precomputed off-line. They can be used by the server to plan aperiodic requests or to speed up the execution of periodic operations. The idle time advanced can be saved as aperiodic capacity at the priority levels of the periodic tasks completed in the latter instance.

A. The Definition of the IPE server

The Improved Priority Exchange (IPE) method discussed here is built on the concept mentioned above. We modify the DPE server in particular by utilizing the idle times of an EDL scheduler. Firstly, a far more efficient server replenishment

policy was obtained. Secondly, the obtained server is no longer periodic and allows to always run as the system's highest priority. As a result, the IPE server is defined as follows: (NB. This is a definition in [1])

- *The IPE server has an aperiodic capacity, which is initially set to 0;*
- *At each instant $t = e_i + kH$, with $0 \leq i \leq p$ and $k \geq 0$, a replenishment of ${\Delta_i}^*$ units of time is scheduled for the server capacity; that is, at time $t = e_0$ the server will receive ${\Delta_0}^*$ units of time (the two arrays/vectors $\mathcal{E}$ and $\mathcal{D}^*$ have been defined in the previous section)."For context, $\mathcal{E} = (e_0, e_1..., e_p)$ which represents the times at which idle times occur, while the second vector $\mathcal{D}^* = ({\Delta_0}^*, {\Delta_1}^*...., {\Delta_p}^*)$ represents the lengths of these idle times";*
- *Regardless of any other deadline, the IPE server priority is always the highest in the system, ;*
- *The IPE server other rules (aperiodic requests and periodic instances executions, exchange and consumption of capacities) are the same as for a DPE server.*

The IPE server has been scheduled with the same set of task shown in fig 3 above in the EDL server, which is shown in fig 4 below. We should also take into account that the server replenishments are set according to the function ${\mathcal{F}^{\text{EDL}}}_{\mathcal{J}}$, which is illustrated in fig 2 above. The arrival of the aperiodic request at $t = 8$ is immediately allocated to one unit of time by the server. Due to past deadline exchanges, the other two units are available at the priority level corresponding to deadline 12 and are allocated immediately after the first. The last one is later allocated at time $t = 12$,when a new unit of time is received by the server In this case, the optimal response time is maintained. As we will see later, the ideal EDL server performs marginally better than IPE in only a few cases. That is, IPE almost always behaves almost optimally.

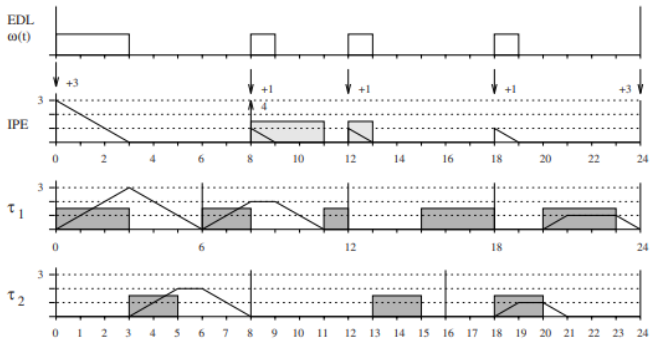


Fig. 4. An Example of Improved Priority Exchange server.[1]

B. IPE Server Schedulability Analysis

For we to analyze the schedulabilty of an IPE server, it is of great importance to firstly define a transformation among schedules similar to DPE server defined schedules. In specific, given a schedule S generated by the IPE algorithm, we construct the schedule $S^{'}$ as follows: (NB. This is a definition in [1])

- for each execution of a periodic instance during a deadline exchange (i.e., a rise in the associated aperiodic capacity) it is delayed until the capacity lowers.
- all other periodic instance executions are left alone as in S.

The server is not replaced by another task in this case. Once more, $S^{'}$ is invariant and well defined, it is dependent only on the periodic task set and not on S. Furthermore, $S^{'}$ is the schedule generated by EDL when applied to the periodic task set.(We compare fig 2 with fig 4). The optimal schedulability is defined by the following Theorem.

"Given a set of n periodic tasks with processor utilization U_p and the corresponding IPE server (the parameters of the server depend on the periodic task set), the whole set is schedulable if and only if"[7]

$$U_p \leq 1$$

(NB. the server allocates automatically the bandwidth $1 - U_p$ to aperiodic requests).

To proof the above theorem:

using *"If"*. The condition for the schedulability of the periodic task set under EDF[4] is enough, As a result, even under EDL, which is a specific implementation of EDF. Now notice that the completion times of the periodic instances in each schedule produced by the IPE algorithm are never larger than the completion times of the corresponding instances in $S^{'}$, which is the schedule of the periodic task set under EDL. That is, no periodic occurrence may be late. The following is the thesis.

"Only If". Simple, because the condition is required even for the periodic task set.[7]

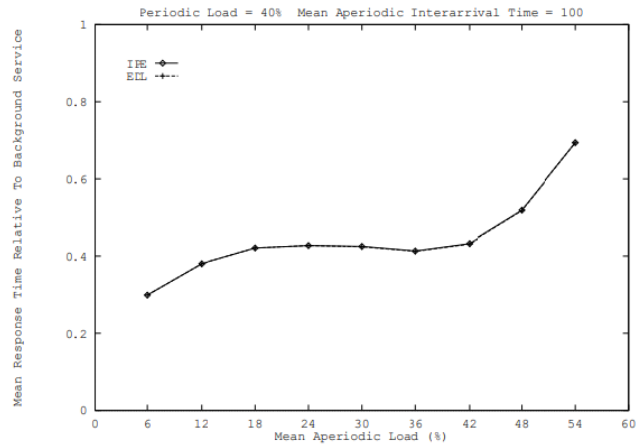
C. The Resource Reclaiming of an IPE server

The same as with the DPE server, the resource reclaiming, that is, reclaiming unused periodic execution time, is possible. When a periodic task is completed, the time saved is added to the associated aperiodic capacity. Again, this behavior has no effect on the system's schedulability. Of course, the cause is the same as with the DPE server.

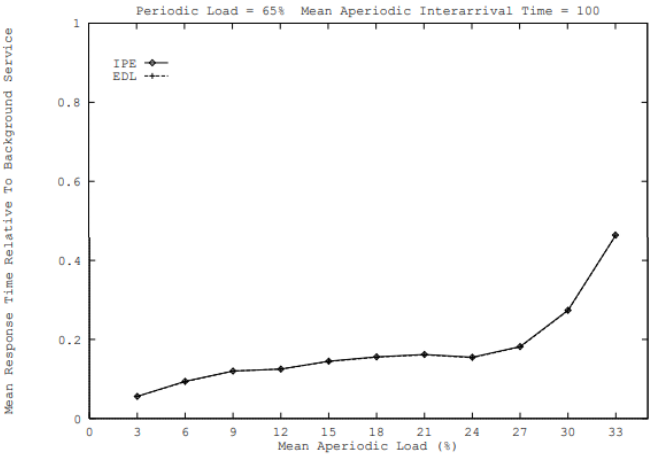
D. Comparing The IPE server and The EDL server, An experiment

We examined the performance of our IPE algorithm to the optimum EDL server method in the first trial. Fig 5 below depicts three graphs that correlate to three distinct periodic loads: low, medium, and high. The aperiodic load was created by taking the mean interarrival time T_a = 100 and adjusting the average aperiodic service time T_s so that the total load encompassed the range from U_p to full processor usage. As the graphs show, there are no significant differences in performance between the two methods for small and medium periodic loads.

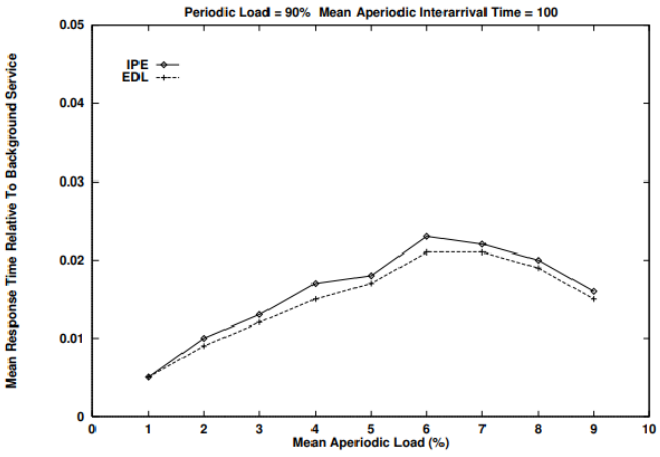
Even with a high periodic load, the difference is so minimal that it may be regarded inconsequential in any practical application. Although IPE and EDL operate relatively similarly,



(a)



(b)



(c)

Fig. 5. IPE and EDL server Comparison.[6]

they differ significantly in terms of implementation difficulty. As described in earlier sections, the EDL algorithm must often recompute the server settings(That is, when new aperiodic request enters the system and all prior aperiodics have been served entirely). This overhead can be prohibitively expensive in terms of CPU time, making the algorithm unsuitable for usage in real applications. For the IPE algorithm, on the other hand, we simply need to compute the server parameters off-line. The implementation overhead is thus the same as for a DPE server at run-time, providing we have enough memory, which is faily acceptable.

To summarise, IPE performs almost as well as EDL, but with far less overhead.

E. IPE server Uppaal Implemenation

The IPE server has been implemented and simulated in uppaal to show the schedulability of an IPE server. Fig 6 below show the simulation; The states, which are the initial idle state where the IPE server aperiodic capacity is set to 0, after the system is ready to schedule(according to the IPE server algorithm definition above, when $k \geq 0$, in this case, x represents K), the system is scheduled as from the IPE server definition, which is regardless of any other deadline, the IPE server is always the highest priority in the system. Then a replenishment unit of time is scheduled for the server capacity. Then the server executes and completes.

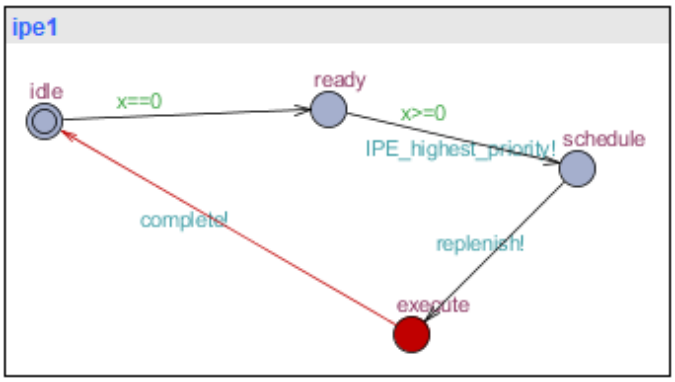


Fig. 6. IPE server Uppaal Implementation

V. Conclusion

We have briefly Introduced Improved Priority Exchange server algorithm in this study. We have also compared the algorithm with two other preceding algorithms, DPE and EDL servers, to have more depth on the performance of this algorithm. We did this by characterising the schedulability of an IPE server with the knowledge of what we know already from the preceding algorithms. A resource claiming approach was also implemented and it was Shown to be effective. We also saw from our brief experiments that, although EDL is optimal, but it has too much overhead, due to its much computation, to be deemed practical. IPE server has shown to achieve same performance but with lesser computational overhead. Although the compared algorithms may require much memory demands, especially when both algorithms

periods of periodic tasks are not related harmonically. With these comparison, a real-time system designer can have basis for efficient system design that incorporates dynamic priorities with aperiodic mechanism service.

VI. Affidavit

I, Abdul-Azeez Olanlokun, thus certify that I wrote the current paper and work entirely by myself, using only the mentioned sources and aids. Other references to the statement and scope are noted by complete information of the publications concerned; words or sections of sentences cited literally are marked as such. The paper and work in a similar or identical format have never been submitted to an examination board or published. This paper had not yet been utilized in another test or as a course performance, even in part.

References

[1] C. Buttazzo, Hard Real-Time Computing Systems, vol. 24. Boston, MA: Springer US, 2011. doi: 10.1007/978-1-4614-0676-1.

[2] M. Spuri and G. Buttazzo, "Scheduling Aperiodic Tasks in Dynamic Priority Systems," The Journal of Real-Time Systems

[3] M. Spuri, G. Buttazzo, nd F. Sensini, "Robust Aperiodic Scheduling under Dynamic Priority Systems," In Proceedings IEEE Real-Time Systems Symposium, pp. 210-219, Pisa, Italy, 5-9 December, 1995.

[4] M. Spuri and G. C. Buttazzo, "Efficient Aperiodic Service under Earliest Deadline Scheduling," In Proceedings IEEE Real-Time Systems Symposium, pp. 2-11, San Juan, Puerto Rice, 7-9 December, 1994.

[5] H. Chetto and M. Chetto. Some results of the earliest deadline scheduling algorithm. IEEE Transactions on Software Engineering, 15(10), 1989

[6] Lehoczky, J.P., and Ramos-Thuel, S., "An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems," Proc. of Real-Time Systems Symposium, 1992, pp. 110-123.

[7] Liu, C.L., and Layland, J.W., "Scheduling Algorithms for Multiprogramming in a Hard real-Time Environment," Journal of the ACM 20(1), 1973, pp. 40-61