Real-Time System Scheduling using Dynamic Priority Exchange Server

A seminar paper on Butazzo's dynamic priority exchange server working with EDF

Emirkan Sali
Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
emirkan.sali@stud.hshl.de

***Abstract*—In the world of real time systems, there are many scheduling options to ensure that the system meets the appropriate timing constraints and works exactly as intended and predicted. The Dynamic priority exchange server is an extension of the priority exchange server based on the Earliest Deadline First algorithm, in which the main function is that priorities of assigned tasks are interchangeable in their runtime.**

***Index Terms*—Real-time, Scheduling, Dynamic, Server, aperiodic, periodic**

I. Introduction

Real-time systems have been a crucial part of society since processor based technology has been introduced to the world. Many examples of application include Industrial, automotive and medical applications and many more [1]. Every computer controlled system needs to perform according to its given function, but in many cases the functions of the systems include time constraints for certain tasks. For these types of applications Real-time systems are required to ensure the execution of given tasks according to the time constraints of the system. If the system reacts too late to a given task, it could have mild or catastrophic consequences depending on the type of system and its use case [1]. Foreseeing and preventing such cases without them happening in the first place, makes predictability the most crucial attribute for real time systems.

For any real-time System, a given task can be seen as a process that needs to be executed by the CPU of the system. To ensure predictability and properly handle all tasks within the given time constraints, an operating system with the proper task scheduling algorithm is needed in the system. When a CPU receives a set of tasks from the system that, it has to be assigned to processing each task in a way that can be determined and controlled [1]. To achieve the proper way of assigning the CPU to the right task at the right time to make sure the system does not fail its time constraints, there are many scheduling algorithms that can be used. In this paper, the main topic will be about the Priority exchange server (DPE), proposed by Spuri and Buttazzo, which is based on the Earliest deadline first algorithm [1].

In an scheduling algorithm, to describe the behaviour of the algorithm and the given taskset, the following notations are introduced [1]:

- Γ is a set of periodic tasks
- τ_i is one task of a periodic taskset
- $\tau_{i,j}$ is the jth instance of task τ_i
- C_i is used to represent the computation time of task τ_i
- T_i is used to represent the period of task τ_i
- ϕ_i represents the current phase of task τ_i
- D_i denotes the relative deadline of a task τ_i
- $d_{i,j}$ is the absolute deadline of task τ_i and its instance j

II. Basics for dynamic priority exchange server

To understand how the dynamic priority exchange server operates, one has to explore the functionality of its basics first. Decomposing DPE, we see:

- DPE is based on the earliest deadline first (EDF) algorithm. [1]
- DPE is a Server in the system, which works in the same way as a periodic task. The polling server serves as its basis. [1]
- DPE can be seen as an extension of the priority exchange server, in which case it is important to understand its functionality first. [1]

A. Earliest Deadline First algorithm

The Earliest Deadline First algorithm is a dynamic algorithm in which every task's priority is directly based on their current deadline [1]. To be more precise, tasks that have earlier deadlines than others will receive a higher priority and will be executed as such. In this case the deadline of a periodic task can be denoted in the same way as in [1] as

$$d_{i,j} = \phi_i + (j-1)T_i + D_i \tag{1}$$

A Schedulability analysis can be performed using the formula

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \tag{2}$$

The bound 1 in this case denotes the CPU being utilized at 100%. Any value below 1 or equal 1 therefore means that

the CPU is able to schedule every task sucessfully under EDF [1]. It is notable that, although EDF is based on deadlines, the schedulability of the algorithm does not depend on its task deadlines. An example by Butazzo [1] of EDF can be seen in figure 1

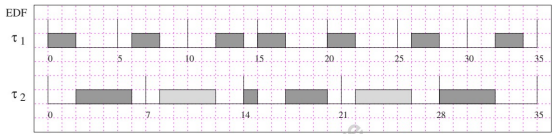

Fig. 1. Example of polling Server approach for a set of two task

B. Polling Server

In many scheduling algorithms the response time for aperiodic requests is not the same as for periodic request, since these requests can arrive at an unpredictable time. A lot of aperiodic requests do not have a tight deadline unlike periodic tasks and are therefore sometimes handled with low priority, [1]. In background Scheduling for example, aperiodic tasks that arrive, are always handled by the First Come First Serve (FCFS) principle, but they are only handled at times, where the CPU has no periodic tasks ready to execute [1].

To improve the response time for aperiodic requests of the system, a server can be introduced. This server runs in the same way as an periodic task and can therefore be scheduled using an algorithm that is optimized in handling periodic tasks [1]. A server also has a computation time C_s, called server capacity and a period T_s like any other periodic task [1]. The polling server is one algorithm that uses a server approach to schedule a set of tasks [1]. In this approach, the server is a periodic task with capacity C_s and period T_s handling aperiodic requests in an system that schedules using Rate Monotic Scheduling (RM) i.e. an algorithm that gives higher priorities to tasks with shorter periods and schedules them based on that [1]. In an example shown by Butazzo [1] one can gain an understanding of how the polling server handles its requests in figure 2

The polling server is a low priority server and will suspend its capacity if there are no aperiodic requests in the time it is active. any aperiodic request will have to wait until the next server period when its capacity is replenished again [1]. In figure 2 one can see how the server capacity is consumed whenever an aperiodic task is ready to execute, denoted by the arrows and their computation time next to them. If the server has no aperiodic tasks to handle, periodic requests will be handled instead and the server suspends its capacity until the next period.

In the case of the polling server, it has a low priority and does not reserve any server capacity during its period, unlike e.g. the Deferrable Server, which can operate at a higher priority than some given tasks and also preserves its capacity when there is no aperiodic request at the start of its period [1].

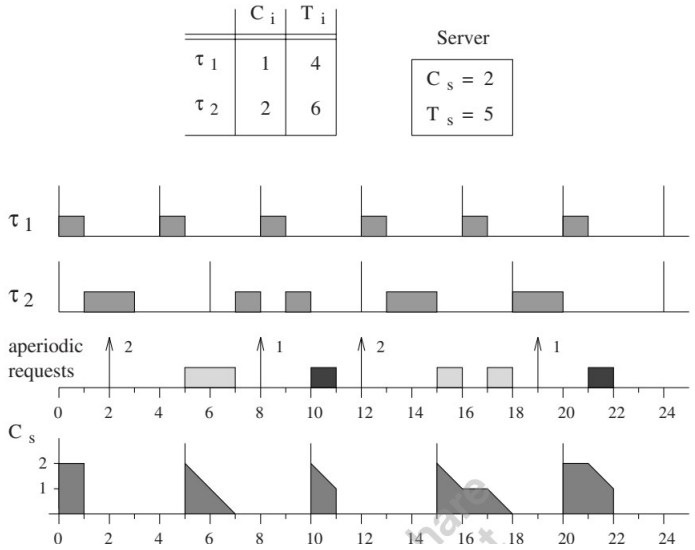

Fig. 2. Example of polling Server approach for a set of two periodic tasks and aperiodic requests

C. Priority exchange Server

The most similar algorithm to DPE is the Priority Exchange (PE) Server proposed by Lehoczky, Sha, and Strosnider [1]. It works in a similar way to the normal polling server in that it is a periodic task, but it is usually a high priority with drastic differences in capacity preservation [1]. Instead of preserving its capacity for itself during the period, PE exchanges its capacity with the execution time of a lower priority task for potentially longer than just one period [1].It means that the capacity of the server basically gets stored in other periodic tasks and it can be transferred back to the server whenever an aperiodic request arrives. The only way in which the server capacity is lost is when the processor sits idle due to lack of periodic tasks. Like any other server, the capacity is set to the normal value at the beginning of every period, but the exchanged capacity will not get lost when other periodic tasks are running. An example of PE by Butazzo [1] is shown in figure 3

at t $= 0$ one can see how the server capacity is consumed and transferred to τ_1 by observing the behaviour of the blank line on the diagram. at t $= 4$, τ_2 arrives and the server capacity is exchanged with the task and consumed from τ_1. at t $= 5$ an aperiodic request with computation time $= 1$ arrives and gets handled by the server using its capacity, which has been replenished at the same time. Note that the capacity stored in τ_2 still remains. At t $= 10$ the server capacity is replenished again and is exchanged with τ_1, because no aperiodic requests are ready to execute at that time and τ_2 is not ready to execute again. the saved capacity of the currenttly running task τ_1 is consumed at t $= 12$, when an aperiodic task with computatio time $= 1$ arrives. after execution, τ_1 continues executing without the exchanged capacity of the server. Lastly at t $= 15$ the server replenishes its capacity again and exchanges it with the current running task τ_2, giving it a capacity of 2 at that

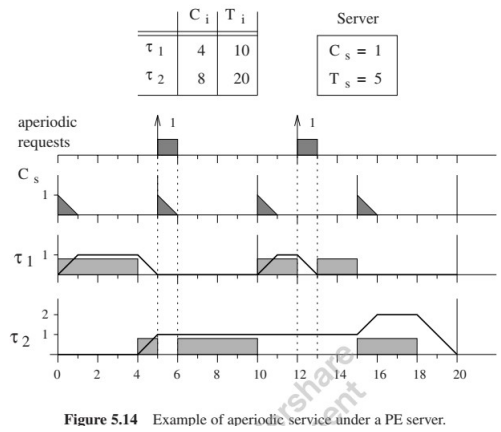

Figure 5.14 Example of aperiodic service under a PE server.

Fig. 3. Example of Priority exchange approach for a set of two periodic tasks and aperiodic requests

point. However, this capacity is lost after execution of τ_2 is finished, because no other tasks are ready to execute.

III. Dynamic priority exchange server

DPE works as an extension of PE in a way that it uses its base functionality, but refines the priority assignment of periodic tasks by using EDF to schedule them. That makes DPE a dynamic approach for PE, because the priorities of a taskset Γ change dynamically during execution, depending on their approaching deadline with respect to the passed time [1].

The algorithm can be described as follows:

- Like any other server it has a period T_s and a capacity T_s [1].
- Each period, the servers aperiodic capacity is replenished to C_S^d with d being the deadline of the current server period [1].
- All periodic tasks have an aperiodic capacity $C_{S_i}^d$, initally set to 0 [1].
- all aperiodic capacities get assigned priorities depending on their current deadline [1].

an example of DPE by Butazzo is visualized in figure 4

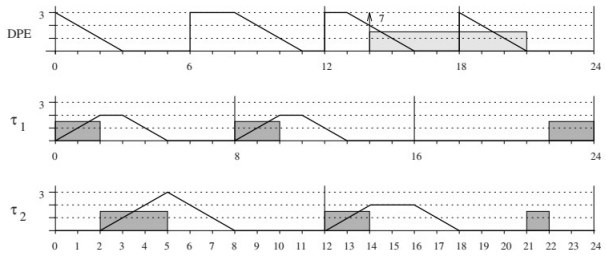

Fig. 4. Example of Priority exchange approach for a set of two periodic tasks and aperiodic requests

IV. Challenges and advantages of DPES

(PLATZHALTER) Advantages and disadvantages

V. Application example

(PLATZHALTER) Applicatione example of DPE

VI. Coclusion

(PLATZHALTER) conclusion of studies on the dynamic priority exchange server

References

[1] G. C. Buttazzo, *Hard real-time computing systems predictable scheduling algorithms and applications*. MTM, 2013.

Statement of authorship

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

Lippstadt, May 16, 2023 .....