Deadline vs. Laxity as a Decision Parameter in Fuzzy Real-Time Scheduling

Mojtaba Sabeghi, Koen Bertels
{sabeghi, k.l.m.bertels}@ce.et.tudelft.nl
Delft University of Technology
Delft, the Netherlands

Mahmoud Naghibzadeh
naghib@um.ac.ir
Ferdowsi University of Mashhad
Mashhad, Iran

Abstract. **Many algorithms have been proposed for scheduling real-time tasks to guarantee the timing requirements of such a system. Each of those algorithms is based on a decision parameter with which the algorithm determines the order of the tasks to be executed. Among those parameters, deadline and laxity are the most important ones which have been employed in the Earliest Deadline First (EDF) and the Least Laxity First (LLF) algorithms respectively [1]. Furthermore, to include the implicit constraints imposed by real world, such as uncertainty and lack of complete knowledge about the environment, dynamicity in the world, bounded validity time of information and other resource constraints, a few fuzzy approaches have been proposed [2, 3]. It was shown that the fuzzy approaches outperform the non-fuzzy approaches. The main purpose of this paper is to study and compare the behavior of a real-time system when the fuzzy scheduler is based on deadline and when it is based on laxity. The obtained results show that deadline is a better parameter to be considered in fuzzy real-time scheduling.**

Keywords: **Real-time Scheduling, Fuzzy Logic, EDF, LLF**

I. Introduction

Scheduling real time systems involves allocation of resources and CPU-time to tasks in such a way that certain performance requirements are met. In real-time systems scheduling plays a more critical role than non-real-time systems because in these systems having the right answer too late is as bad as not having it at all [1]. Such a system must react to the requests within a fixed amount of time which is called deadline.

Real-time tasks can be classified as periodic or aperiodic. A periodic task is a kind of task that occurs at regular intervals, and aperiodic task occurs unpredictably. The length of the time interval between the arrivals of two consecutive requests in a periodic task is called period.

There are a plenty of real-time scheduling algorithms that are proposed in the literature. Each of these algorithms bases its decision on certain parameter while attempting to schedule tasks to satisfy their time requirements. Some algorithms use parameters that are determined statically such as the Rate Monotonic algorithm that uses the request interval of each task as its priority [4, 7]. Others use parameters that are calculated at run time. Laxity and deadline are among those parameters that are the most considered. Laxity says the task execution must begin within a certain amount of time while deadline implies the time instant at which its execution must be completed [2, 5, 6].

The main purpose of this paper is to study and compare the behavior of a real-time system when the fuzzy scheduler is based on deadline and when it is based on laxity. The rest of the paper is organized as follows. In Part II we will give an introduction to fuzzy inference process. Part III covers the algorithms and part IV presents the experimental results. Conclusion and future works are debated in Sections V.

II. 2. Fuzzy Inference Systems

A general fuzzy inference system consists of three parts. A crisp input is fuzzified by input membership functions and processed by a fuzzy logic interpretation of a set of fuzzy rules. This is followed by the defuzzification stage resulting in a crisp output [8].

There are a number of different ways to implement the fuzzy inference engine. Among the very first such proposed techniques is that due to Mamdani [10], which describes the inference engine in terms of a fuzzy relation matrix and uses the compositional rule of inference to arrive at the output fuzzy set for a given input fuzzy set. The output fuzzy set is subsequently defuzzified to arrive at a crisp control action.

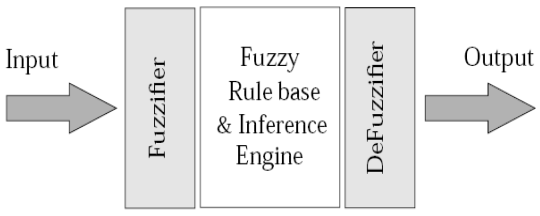

Fig 1. Block diagram of fuzzy inference process

The inference engine is based on a collection of logic rules in the form of IF-THEN statements, where the IF part is called the "antecedent" and the THEN part is called the

"consequent". An example of fuzzy IF-THEN rules is: IF frequency is high then priority is low, which frequency and priority are linguistics variables and high and low are linguistics terms. Typical fuzzy inference systems have dozens of rules. These rules are stored in a knowledgebase.

There are two common inference processes [8]. First is called Mamdani's fuzzy inference method proposed in 1975 by Ebrahim Mamdani [10] and the other is Takagi-Sugeno-Kang, or simply Sugeno, method of fuzzy inference Introduced in 1985 [11]. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant but Mamdani's inference expects the output membership functions to be fuzzy sets.

III. FUZZY ALGORITHMS

MFDF, MFLF, FGEDF, FPEDF, FGMLF and FPMLF algorithms has been considered here. Each of the algorithms is as follows:

Algorithm MFDF

Loop
1. For each task T, feed its external priority and deadline into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (laxity, deadline, etc)
End loop

Algorithm MFLF

Loop
1. For each task T, feed its external priority and laxity into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (laxity, deadline, etc)
End loop

The algorithm for the MFDF is similar to the MFLF with laxity replaced by deadline.

Algorithm FGEDF

Loop
For each CPU in the system do the followings:
1. for each ready task T (a task which is not running), feed its external priority and deadline into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (deadline, etc)
End
End loop

Algorithm FGMLF

Loop
For each CPU in the system do the followings:
1. for each ready task T (a task which is not running), feed its external priority and laxity into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (deadline, etc)
End
End loop

FGMLF is much the same with FGEDF just by replacing the word deadline by laxity.

Algorithm FPEDF for each CPU

Loop
1. For each ready task T (a task which have not been run on another CPU), feed its external priority and deadline into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (deadline, etc)
End loop

Algorithm FPMLF for each CPU

Loop

1. For each ready task T (a task which have not been run on another CPU), feed its external priority and laxity into the inference engine. Consider the output of inference module as priority of task T.
2. Execute the task with highest priority until an scheduling event occurs (a running task finishes, a new task arrives)
3. Update the system states (deadline, etc)

End loop

FPMLF is much the same with FPEDF just by replacing the word deadline by laxity.

For a detail explanation of each algorithm refer to [2,3,9,12].

IV. Simulation Results

Performance metrics, which are used to compare different algorithms, must be carefully chosen to reflect the real characteristics of a system. These metrics are as follows. Response time, which is defined as the amount of time a system takes to react to a given input, is one of the most important factors in most scheduling algorithms. Number of missed deadlines is an influential metric in scheduling algorithms for soft real-time systems. When task preemption is allowed, another prominent metric comes into existence and that is the number of preemptions. Each of preemptions requires the system to perform a context switching which is a time consuming action. CPU utilization is also an important metric because the main goal of a scheduling algorithm is to assign and manage system resources so that a good utilization is achieved. Yet another metric, which is considered in our study, is the number of missed deadlines from the class of highest priority tasks. This corresponds to the external priority being *very high*.

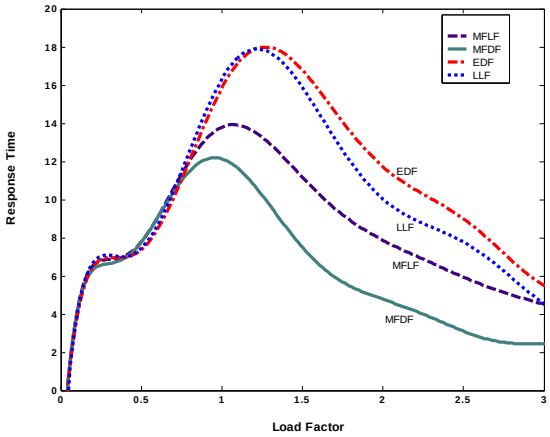

Fig.2. Response time in overloaded conditions

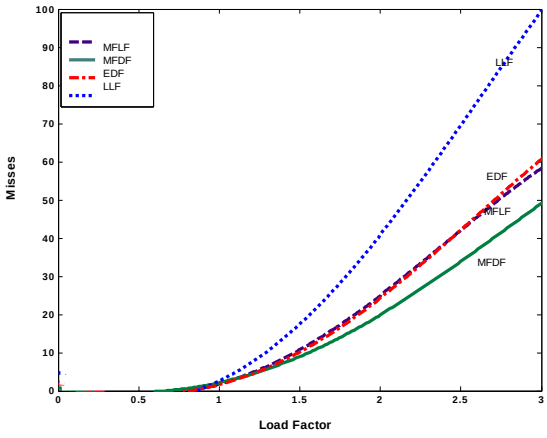

Fig.3. Number of Misses

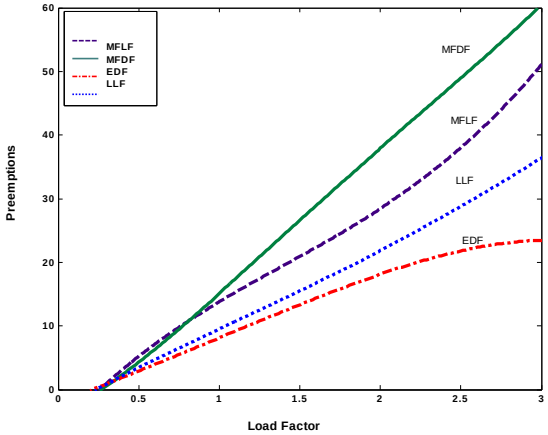

Fig.4. Number of Preemptions

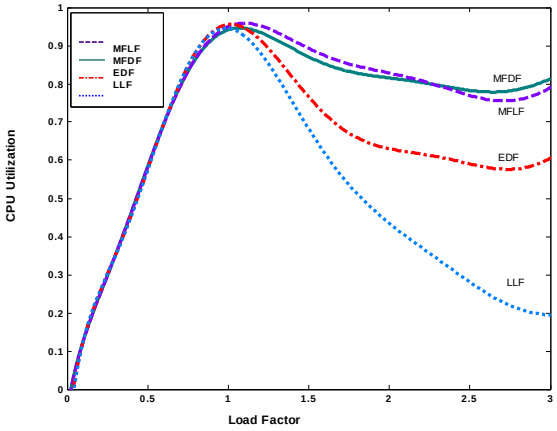

Fig.5. CPU Utilization

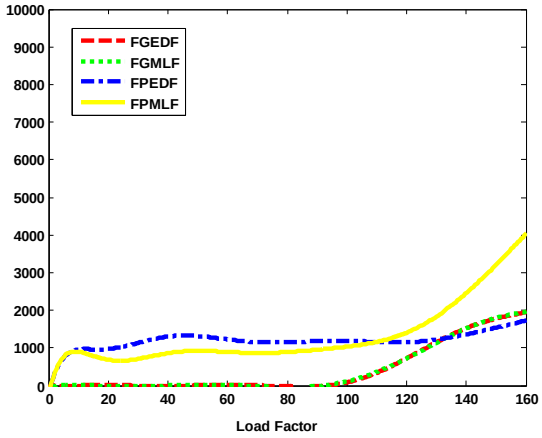

Fig.6. Number of misses for 128 CPUs

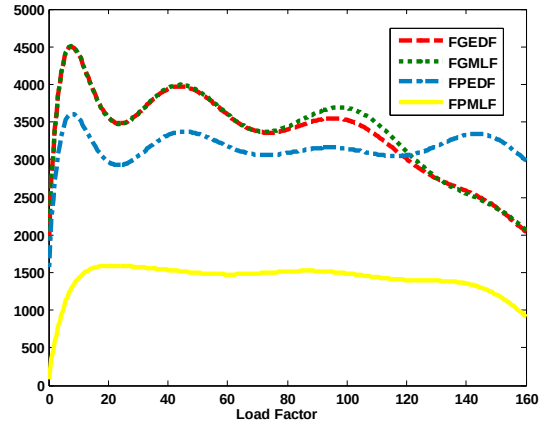

Fig.7. Number of preemptions for 128 CPUs

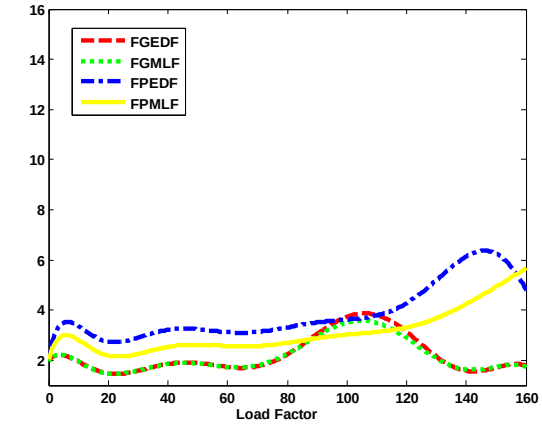

Fig.8. Average response time for 128 CPUs

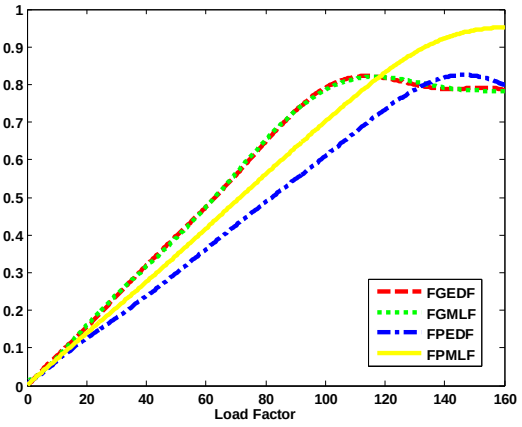

Fig.9. Average CPU utilization for 128 CPUs

As it is shown in the figures 2 to 9, algorithms based on laxity have a better CPU utilization both for multi processor and single processors. But it misses more tasks than deadline based algorithms.

Among the four algorithms FGEDF and FGMLF nearly achieve the same performance in all situations and all metrics. FPMLF performs poorly in number of misses and also average response time, but its performance on CPU utilization and also number of preemption is much better than the others especially when the number of CPUs increase.

V. Conclusion

This paper compared fuzzy algorithms based on deadline and laxity. As it was shown, using deadline as a fuzzy parameter in real-time scheduling is more promising than laxity. Also, it seems that partitioning approach almost outperforms global approach for multiprocessor real-time scheduling.

References

[1] Liu C. L., Layland J. W., Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. Journal of the ACM, 20(1):46-61, 1973.

[2] Sabeghi M., Naghibzadeh M., Taghavi T., "Scheduling Non-Preemptive Periodic Tasks in Soft Real-Time Systems using Fuzzy Inference", *9th IEEE International Symposium on Object and component-oriented Real-time distributed Computing* (ISORC), April 2006

[3] Sabeghi M., Deldari H., "A Fuzzy Algorithm for Scheduling Periodic Tasks on Multiprocessor Soft Real-Time Systems", *IASTED International Conference on Modeling and Simulation*, May 2006

[4] Ramamritham K., Stankovic J. A., Scheduling algorithms and operating systems support for real-time systems,

Proceedings of the IEEE, Vol. 82(1), pp55-67, January 1994.

[5] Goossens J., Richard P., Overview of real-time scheduling problems, Euro Workshop on Project Management and Scheduling, 2004

[6] Hong J., Tan X., Towsley D., A Performance Analysis of Minimum Laxity and Earliest Deadline Scheduling in a Real-Time System, *IEEE Trans. on Comp.*, Vol. 38, No. 12**,** Dec. 1989

[7] Sha, L. and Goodenough, J. B., Real-Time Scheduling Theory and Ada, IEEE Computer, Vol. 23, No. 4, pp. 53-62 (April 1990).

[8] Wang Lie-Xin, A course in fuzzy systems and control, Prentice Hall, Paperback, Published August 1996.

[9] Andersson B., Jonsson J. Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition, Seventh International Conference on Real-Time Computing Systems and Applications (RTCSA'00), 2000

[10] Mamdani E.H., Assilian S., An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies, Vol. 7, No. 1, pp. 1-13, 1975.

[11] Sugeno, M., Industrial applications of fuzzy control, Elsevier Science Inc., New York, NY, 1985.

[12] Sabeghi M., Naghibzadeh M., Taghavi T., A Fuzzy Algorithm for Scheduling Soft Periodic Tasks in Preemptive Real-Time Systems, International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE), 2005