Real-time scheduling for embedded systems using enhanced EDF

Steve Senviel, Qin Lu-Stone

COEN 283, Spring 2010

Santa Clara University

Abstract

The need for a real-time scheduling algorithm becomes evident with the increase demand for embedded applications. This paper reviews past researches, addresses the problem of regular EDF setup, and propose an enhanced EDF algorithm. In conclusion, enhanced EDF can provide better performance for real-time applications.

1. Introduction

From digital camera to mobile phones, embedded systems can be found in every aspect of our life. In recent years, embedded systems have become more and more resource intensive because of the increase demand for real-time and quality-of-service(QoS) requirements. There is a urgent need to design a system that can meet the functional and timing requirements.

This paper serves as a team project for COEN283, Operating Systems, at Santa Clara University in Spring 2010. We chose this topic for three reasons: 1) processor scheduling is the core topic of all modern operating systems. Without efficient scheduling algorithm, no operating system can function to its best extent; 2) searching for a better process algorithm is an on-going topic in the operating systems research field. Process scheduling plays a key function in all embedded devices, especially mobile devices.

There have been many researches focus on scheduling deadlines, priority scheduling algorithms and energy consumption. However, none offers a solution that will satisfy both the need of scheduling deadlines and the need of real-time application. In the following sections, we will show you how to set up a system that works for real-time embedded application using modified EDF.

The remainder of this paper will cover the theoretical bases of the problem and recent research, hypothesis, methodology we used to solve the problem, code implementation, data analysis and conclusions.

2. Theoretical bases

a. Definitions

Embedded systems - Computer systems that are designed to perform a small or specific set of tasks, often in. At the bare minimum embedded systems are single CPU systems with a relatively small and fixed amount of memory but they can also include other IC's or I/O devices which interface with the main CPU.

Scheduling - A function of an operating system which manages which process will get to run next on the CPU, and how long processes or tasks get to run on the CPU. On embedded systems real-time processes must execute before a certain time and sometimes they are performing atomic operations, which cannot be disrupted once they start. These types of processes create the necessity to use non-preemptive scheduling algorithms and the need to choose processes in an order that will optimize system efficiency.

Real time - Used to characterize a time constraint of a task of process. Embedded systems sometimes have to be able to execute real time tasks depending on the use of the embedded system.

EDF - a dynamic scheduling algorithm used in real-time operating systems. It assigns priority to process that has the earliest deadline.

b. Related researches:

There are several known real-time algorithms for embedded systems. They are Earliest Deadline First (EDF), Least laxity first (LLF) and Rate Monotonic Scheduling (RMS).

In EDF, tasks are scheduled by the order of deadline. In LLF, tasks are scheduled by the order of laxity. EDF and LLF do not work very well in situations that soft real-time applications compete with higher priority applications.

In RMS, tasks are scheduled by static priority that is determined by duration. The shorter the job duration is, the higher its priority is. It guarantees time restraints up to 70% CPU load. When the load is above 70%, it does not support dynamic systems very well.

Recent research work includes Constant Bandwidth server that changes task deadlines, R-EDF that is based on task utilization and performs well in mix environment of soft real-time applications and priority driven applications. However R-EDF does not work in hard real-time situation.

c. Our solution

Based on EDF, we implement Enhanced R-EDF that allows a task to exit its overturn state and execute in the available time.

3. Hypothesis

With the same system setup, same number of jobs running, we believe our solution will provide lower percentage of tasks that miss deadline.

4. Methodology

We use MOSS scheduling simulator to simulate enhanced EDF scheduling algorithm. Designed by Alex Reeder, MOSS scheduling simulator provides statistical analysis for a mix of process tasks in a simulated environment. It is written in JAVA, which makes it platform independent. the program uses a configuration file to define the necessary inputs for the simulation. Originally the configuration file allows the user to specify the number of processes to run, an average runtime and a standard deviation for the process run times and the amount of time a given process will run before it will need to block the CPU for I/O. for our experiment we have modified this file and the existing code to allow for a class specification for each process. the class is an indicator of how I/O bound the process is with class 1 processes being the least I/O bound and class 4 processes being the most I/O bound. the program creates two output files, one

is a summary of the results of the algorithm that was run and the other is log file that shows a trace of when process are running, blocking for input or have finished.

5. Implementation

In order to create the effect of random cpu run times for each process we need made a slight modification to the what was given to us with the original MOSS simulator code. the simulator uses the built in JAVA random number generator and uses a system time in milliseconds as the seed to re-initialize the generator for the cpu runtime generation for each process. This was not problem when the simulator was written in 2001, but processors today tend to use a finer granularity that 1ms and this was causing us to get repetitions for the runtimes of the processes. We found a way around this problem by making method that generated the random runtime sleep for 1ms before it returned. this allowed us have a different seed for the next time this method was called.
We chose to implement the scheduler as a queue that accepts processes at a constant arrival rate. when a process arrives in the queue it is stamped with an arrival time. the deadline of each processes is given as a function of its arrival time, its process class and its I/O blocking parameters
6. Data analysis and discussion
To generate our data for analayis of our algorithm we created a configuration file that defined 32 process with varying classes, a mean runtime of 500 time units and a standard deviation of 250 time units. as a base case four our algorithm we used the basic EDF algorithm which scans the entire scheduling queue for a the available process with the earliest deadline.

7. Conclusions and recommendations

This paper proposed an enhanced EDF algorithm that meets the needs of real-time applications. Enhanced EDF however did not display significant improvement on missed deadlines over the regular EDF. We suspect there may be some idle time during switches of processes that causes delay.

Bibliography

Development of Scheduler for Real Time and Embedded System Domain

Rao, M.V.P.; Shet, K.C.; Balakrishna, R.; Roopa, K.; Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on , 1-6, 2008

Andrew S. Tanenbaum, Modern Operating Systems, 2008

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

R-EDF: a reservation-based EDF scheduling algorithm for multiple multimedia task classes, Wanghong Yuan; Nahrstedt, K.; Kihun Kim; Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE Digital Object Identifier: 10.1109/RTTAS.2001.929881 Publication Year: 2001, Page(s): 149 - 154

ER-EDF: A QoS Scheduler for Real-Time Embedded Systems

Matschulat, D.; Marcon, C.A.M.; Hessel, F.; Rapid System Prototyping, 2007. RSP 2007. 18th IEEE/IFIP International Workshop on Digital Object Identifier: 10.1109/RSP.2007.22 Publication Year: 2007, Page(s): 181 - 188

http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling