A QoS Scheduler for Real-Time Embedded Systems

David Matschulat, César A. M. Marcon, Fabiano Hessel
PPGCC - FACIN – PUCRS - Av. Ipiranga, 6681, Porto Alegre, RS – Brazil
Fabiano.Hessel@pucrs.br

Abstract

The increasing demand for real-time embedded applications makes evident the need for end-to-end Quality of Service (QoS) provisioning. In order to achieve the end-to-end QoS, we propose the implementation of the control and management of QoS mechanisms in the operating system scheduler. A new scheduling algorithm, named ER-EDF, is proposed and compared to previous scheduler solutions. This approach was validated through a set of benchmarks and we conclude that ER-EDF adds performance and simplified hard real-time support to real-time embedded applications.

1. Introduction

Real-Time Operating Systems (RTOS) services and mechanisms (e.g. scheduling) with QoS support emerged to provide predictability to the critical systems. However, the current generation of RTOS schedulers lacks adequate support for applications with stringent QoS requirements. Since processing and communication requirements are distinct for each media type, different QoS guarantees are necessaries to maintain synchronization characteristics, temporal constraints, and reliability of an application.

The key issues of this paper is to discuss and propose an adequate support for QoS provisioning and service adaptability that can be built in a general purpose RTOS. In this sense, we present a new scheduling algorithm for QoS provisioning (ER-EDF). In this paper, we mainly focus on the QoS provisioning for hard real-time tasks.

In order to validate the proposed approach, three algorithms were implemented in an RTOS: EDF, *Reservation* EDF (R-EDF) and *Enhanced* R-EDF (ER-EDF). The main issue of the ER-EDF is the performance enhancements and support for processing reservation for hard real-time tasks.

The remaining of this paper is organized as follows. Section 2 presents the related work. Sections 3, the basic concepts of job and task model are explained. Section 4 presents the R-EDF algorithm and its limitations. Section 5 presents the new algorithm ER-EDF. Section 6 shows the implementation and experiments created to validate the proposed algorithm. Finally, Section 7 concludes this work.

2. Related work

Deng [1] proposed a scheduling scheme for hard real-time applications in open environment. However, these algorithms usually do not work well where soft real-time applications coexist with best-effort applications. Abeni and Buttazzo [2] introduced the Constant Bandwidth Server (CBS), which schedules tasks based on budget reservation. CBS restricts the execution of tasks to its budget to protect other tasks, thus allowing unnecessary deadline misses.

Zhu [3] proposed the Diff-EDF scheduler, which offers guarantees to tasks by changing a task's deadlines based on its desired miss-rate. Diff-EDF lacks support to multiple classes and hard real-time tasks. SMART [4] and Rialto [5] allow applications to specify real-time requirements for a computation unit. These approaches may incur a large overhead, since real-time applications usually contain a lot of code blocks with timing constraints and it is necessary to specify time constraints for each individual code block.

Yuan [6] introduced the R-EDF algorithm, which targets the mix of soft real-time applications and best-effort applications. However, the algorithm can deliver unexpected results for soft real-time applications generating undesirable delays. This work improves R-EDF algorithm by hard real-time supporting and a better overall performance of the application.

3. Job and Task Models

Models of real-time systems may use the concepts of task and job to represent the behavior of applications. A task is a part of an application, since

0-7695-3117-2/08 $25.00 © 2008 IEEE
DOI 10.1109/ISQED.2008.94

that an application can be seen as a set of tasks. A job computes part of a task, having a release time and deadline.

Together with the release time and deadline, a job has the processing time **P** and the relative deadline **R**, which, for this work, it is also considered the period of the task. The utilization **θ** of a job **J** is **θ(J)** = **P/R**. The task **T** is composed by a set of jobs ($\mathbf{T} = \{\mathbf{J_1}, \mathbf{J_2}, ..., \mathbf{J_n}\}$, where $\mathbf{n} \geq 1$). In a task **T** with **n** jobs, the utilization of the task is $\theta(\mathrm{T}) = \frac{\sum_{i=1}^{n} \theta(\mathbf{J_i})}{\mathbf{n}}$. The current job list (**CJL**) of a task is the set of jobs that have already been released but not yet completed. That is, $\mathbf{CJL(T)} = \{\mathbf{J_s}, \ldots, \mathbf{J_m}\}$, where $\mathbf{J_m}$ is the latest released job.

4. R-EDF Concepts and Limitations

R-EDF is based on the EDF, and proposes to add QoS to task scheduling. It is accomplished by reserving the processing time via parameterization. R-EDF classifies five types of tasks:

- **Periodic Constant Processing Time (PCPT)** jobs have constant processing time and relative deadline, resulting in constant utilization.
- **Events** are a special kind of PCPT with only one job.
- **Periodic Variable Processing Time (PVPT)** jobs have constant relative deadline and variable processing time.
- **Aperiodic/Sporadic Constant Utilization (ASCU)** jobs have arbitrary relative deadlines and processing time.
- **Best-effort** tasks have no timing restrictions, but should not starve.

Utilization **θ** and peak utilization **ψ** are defined, as the average utilization of all jobs of a task and the maximum utilization among all jobs of a task, respectively. Each task reserves the processing time for all its jobs at the beginning of the task, based on **θ** for soft real-time tasks and **ψ** for hard real-time tasks. When a job exceeds its reservation limit, it enters in the overrun state. The job returns to the ready state when the next release time of the task comes.

Therefore, a system with **M** processor has capacity **M**. R-EDF statistically multiplexes the processor capacity between real-time and best-effort tasks. The time-sharing capacity $\mathbf{C_{TS}}$ is the unreserved capacity, which is shared among all best-effort tasks. $\mathbf{C_{TS}}$ has a lower bound **β**, such that $\mathbf{C_{TS}} \geq \beta$, to protect best-effort tasks from starvation. Real-time capacity $\mathbf{C_{RTp}}$ and peak capacity $\mathbf{PC_{RTp}}$ of a processor **p** ($1 \leq \mathbf{p} \leq \mathbf{M}$) are, respectively, the sum of the utilizations of the tasks and the sum of peak utilization of tasks bound to a processor. That is, $\mathbf{C_{RTp}} = \sum_{i=1}^{m} \theta(\mathbf{T_i})$ and $\mathbf{C_{RTp}} = \sum_{i=1}^{m} \theta(\mathbf{T_i})$, where $\mathbf{T_i}$ ($1 \leq \mathbf{i} \leq \mathbf{m}$) are real-time tasks bound to a processor **p**. The system is classified as being real-time *overloaded* if $\mathbf{PC_{RTp}} > 1$, or $\sum_{p=1}^{M} \mathbf{PC_{RTp}} > \mathbf{M} - \beta$ for the whole system. Otherwise, the system is *under loaded*. Analyzing the R-EDF algorithm's behavior, a limitation was found: the restrictive reservation. This limitation appears when a job uses all its reserved time and enters in the overrun state. In the most cases would be time available to complete the execution of the job in the period (expanding automatically the reservation). In the proposed algorithm (ER-EDF) we solve this issue by allowing tasks to execute in the extra available time (see Section 5). In addition, R-EDF does not support hard real-time tasks, since it assumes that a job can miss its deadline when the reservation is reached. In this algorithm, if a task enters in the overrun state, it will miss its deadline.

5. The ER-EDF Algorithm

Enhanced R-EDF (ER-EDF) is an improvement of R-EDF. It was conceived to improve the QoS delivered to soft real-time tasks and provide reservation for hard-real time ones.

The hard real-time reservation is accomplished with the establishment of reservations based on the peak utilization **ψ** of a task. This allows the scheduling of hard real-time, soft real-time and best effort tasks in the same system.

5.1. Admission Control

Changes in the admission control were introduced to effect the proposed alterations. At the creation time, each task informs the scheduler if it is a soft or hard real-time task. The admission control algorithm is presented below.

```
Step 1: real-time capacity $C_{RTp}$ and peak real-time capacity
$PC_{RTp}$ of each processor p are set to 0 and the time-sharing
capacity $C_{TS}$ is set to M.
Step 2: A real-time task with utilization $\theta$ and peak utilization $\psi$
requests reservation:
    If the task is hard real-time then (reserve using $\psi$)
```

```
        **If** the time-sharing capacity can be reduced to admit
           this task $C_{TS} - \psi > \beta$, and a processor **p** can fulfill the
           requirement $C_{RTp} + \psi \leq 1$ **then**
                Task is bound to the processor **p**, with: $C_{RTp} = C_{RTp} + \psi$; $PC_{RTp} = PC_{RTp} + \psi$; $C_{TS} = C_{TS} - \psi$
        **Else**
                Task is rejected.
    **Else** (reserve using utilization $\theta$):
        **If** the time-sharing capacity can be reduced to admit
           this task $C_{TS} - \theta > \beta$ and a processor **p** can fulfill the
           requirement $C_{RTp} + \theta \leq 1$ **then**
              Task is bound to the processor **p**, with: $C_{RTp} = C_{RTp} + \theta$; $PC_{RTp} = PC_{RTp} + \psi$; $C_{TS}$ = CTS - $\theta$
           **Else**
              Task is rejected
**Step 3:** **If** a real-time task with utilization $\theta$ and peak utilization
$\psi$, bound to a processor **p**, releases its reservation, **then**:
    **If** the task is hard real-time, **then**
         $C_{RTp} = C_{RTp}-\psi$; $PC_{RTp} = PC_{RTp}-\psi$; $C_{TS} = C_{TS}+\psi$.
    **Else**
         $C_{RTp} = C_{RTp} - \theta$; $PC_{RTp} = PC_{RTp} - \psi$; $C_{TS} = C_{TS} + \theta$.
    **End if**
```

5.2. Scheduling

ER-EDF incorporates modifications to allow better use of the processing capacity. These modifications are conceived to allow a task to exit its overrun state and execute in the available time. The modifications include:

1. Forbid a task to enter in the overrun state when there is not any real-time task ready;

2. At the end of a job, remove the task with the earliest deadline from the overrun state if no other real-time task is ready to execute.

Like its predecessor, ER-EDF only activates the overrun protection mechanism when the system is overloaded. Consequently, ER-EDF has analogous behavior as the EDF algorithm, when the system is under loaded. The ER-EDF algorithm is described next.

```
**Step 1:** Selection a task for execution.
    **If** any real-time task is ready, **then**
          Select one whose latest released job has the earliest
            deadline and execute jobs in the CJL in order;
    **Else if** there is a task in the overrun state, **then**
          Select the task in the overrun state whose latest
          released job has the earliest deadline, put it in the
          ready state and execute jobs in the CJL in order.
    **Else**
          Invoke the best-effort task scheduler.
**Step 2:** The scheduler waits until the next time unit.
    **If** a running task finishes all its jobs, **then**
          It enters the waiting state;
    **Else if** the system is overloaded and the CJL of the current
      task is not empty and the task used all its reserved time,
      **then**
          **If** there is any real-time task ready **then**
                Current task enters the overrun state.
          **Else if** the ran utilization of the current task is greater
             or equal than (1- $\beta$), **then**
                It enters the overrun state.
          **Else**
                It continues to execute.
       Check all tasks for reached release times and set them to
       the ready state.
**Step 3:** Go to step 1.
```

6. Implementation/Experiments

The Spartan-3 Starter Board [7], together with MIPS soft-core processor, was used to validate the proposed algorithms. The operating system used is EPOS[1].

The first experiment has 4 tasks. Each task has 500 jobs to execute simultaneously. The first three are PCPT tasks ($\theta = \psi$). Thus, PCPT tasks have similar behavior to tasks marked as hard real-time.

Table 1 presents the parameters used to generate the experiment data. Task 2 is the only one marked as hard real-time. Task 4 is a PVPT task where each job receives a generated utilization based on a linear distribution, where the minimum is 10% and the maximum is 42%.

Table 1: Parameters for first experiment

Task	1	2	3	4
Jobs	500			
Period	50ms			
Best-effort (β)	0%			
Distribution	Cst	Cst	Cst	Linear
Distribution Param.	26%	21%	26%	10-42%
Utilization (θ)	26%	21%	26%	27%
Peak Utilization (ψ)	26%	21%	26%	42%
Total Utilization (θ)	100%			
Total Peak Util. (ψ)	115%			
Total Reservation	100%			

The deadline miss results for the four tasks of the execution are presented in Figure 1. In this experiment, the hard real-time parameter was disabled in R-EDF and ER-EDF to verify the similar behavior for PCPT tasks. Even though only the second task is marked as hard real-time, all first three behave similarly, losing 0% of its deadlines. In the execution of the EDF algorithm, all tasks miss deadlines. R-EDF and ER-EDF present no deadline miss for PCPT tasks, which is therefore compensated in the PVPT task. However, ER-EDF presents lower deadline miss rate compared to R-EDF.

[1] Available in http://epos.lisha.ufsc.br/

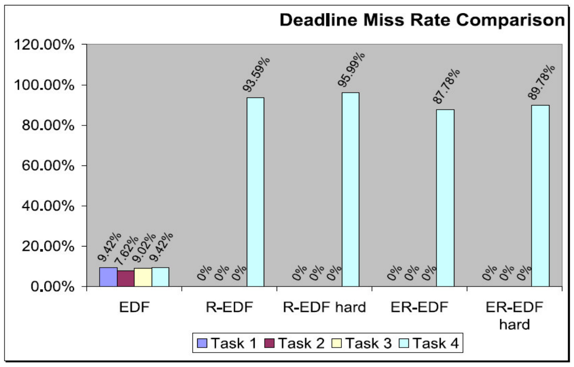

Figure 1: Deadline miss comparison

The next experiment shows two tasks with different periods executing in an overloaded environment. The first task is a hard real-time PCPT with constant utilization of 50%. The second is a PVPT with numbers generated by a linear distribution. Table 2 shows the parameters for the second experiment.

Table 2: Parameters for second experiment

Task	1	2
Jobs	500	250
Period	50ms	100ms
Best-effort (β)	0%	
Distribution	Constant	Linear
Distribution Param.	50%	20-75%
Utilization(θ)	50%	49%
Peak Utilization(ψ)	50%	75%
Total Utilization (θ)	99%	
Total Peak Util.(ψ)	125%	
Total Reservation	99%	

Figure 2 compares EDF, R-EDF and ER-EDF showing their deadline miss rate EDF presents an undesirable behavior showing considerable deadline miss rate for both tasks. R-EDF shows the first task correctly being treated as hard real-time with 0% deadline miss. ER-EDF shows analogous execution to R-EDF. However, ER-EDF presents an improvement of 30% on the second task over R-EDF.

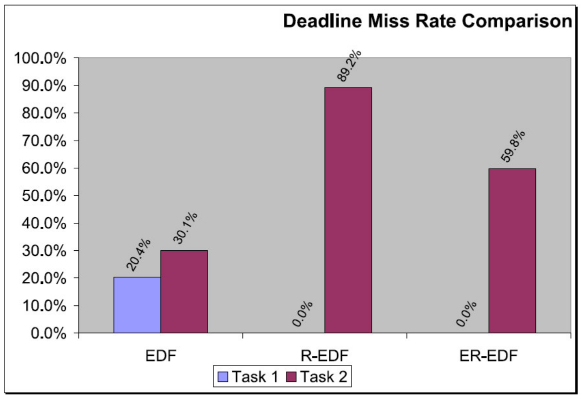

Figure 2: Deadline miss comparison

7. Conclusions

This work introduced a new real-time scheduler algorithm to provide quality of service to applications. The new algorithm – *Enhanced* R-EDF – is based on R-EDF, a multiclass real-time scheduler. R-EDF presents some limitations that are overcome by the new algorithm. In addition, the support for hard real-time tasks was added, which is fundamental to applications that require great responsiveness, and allows the existence of hard real-time, soft-real time and best effort tasks in the same system.

ER-EDF showed significant improvement over its predecessor R-EDF. The addition of hard real-time support allows developers to parameterize the application to fulfill application's real-time requirements. However, the enhancement of real-time execution costs to the best-effort tasks more starvation time.

8. References

[1] Z. Deng and J. Liu. Scheduling Real-Time Applications in an Open Environment. **IEEE Real-Time Systems Symposium**, p. 308-319, 1997.

[2] L. Abeni and G. Buttazzo. Resource Reservation in Dynamic Real-Time Systems. **Real-Time Systems**, v. 27, n. 2, pp. 123-167, 2004.

[3] H. Zhu et al. Diff-EDF: A Simple Mechanism for Differentiated EDF Service. **IEEE Real Time Technology and Applications Symposium**, pp. 268–277, 2005.

[4] J. Nieh and M. Lam. The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications. **SOSP**, pp. 184–197, 1997.

[5] M. Jones et al. An Overview of the Rialto Real-Time Architecture. **ACM SIGOPS European Workshop**, pp. 249-256, 1996.

[6] W. Yuan, K. Nahrstedt, and K. Kim. R-EDF: A reservation-based EDF scheduling algorithm for multiple multimedia task classes. **IEEE Real Time Technology and Applications Symposium**, pp. 149-154, 2001.

[7] Xilinx, Inc. Spartan-3 Starter Kit Board - User Guide, 2005. Available at www.xilinx.com/bvdocs/userguides/ug130.pdf.