Journal of
Information Technology & Software Engineering

Chetto, J Inform Tech Softw Eng 2015, 5:3
DOI: 10.4172/2165-7866.1000e124

Editorial Open Access

Graceful Overload Management in Firm Real-Time Systems

Maryline Chetto*

IRCCyN (Institut de Recherche en Communications et Cybernétique de Nantes), University of Nantes, Nantes, France

Abstract

Real-time embedded systems have to provide the highest Quality of Service despite possible processing overloads. In such systems, programs are characterized by upper bounds on finishing times and the QoS is assessed by the ratio of successful deadlines. In this article, we deal with this issue. We focus on single processor architectures in the framework of firm real-time applications that accept deadline missing under some specified limits. Tasks are assumed to be periodic. We present a novel model for tasks which is called BGW model. It is drawn from two approaches respectively known as the skip-over model and the Deadline Mechanism. We propose specific dynamic priority schedulers based on EDF (Earliest Deadline First) for BGW task sets and we briefly report the results of a simulation study.

Introduction to Real-Time systems

In real-time environments, tasks have to complete by their deadlines. As we restrict our attention to mono-processor systems, a scheduling algorithm aims to determine which task is to execute on the processor [1,2]. On-line scheduling algorithms have been designed under non-overloaded conditions. We say that a real-time system is overloaded when it is impossible to schedule it on the processor such that all the tasks meet their deadlines. Classical online priority driven schedulers perform poorly as compared to offline clairvoyant ones in overloaded conditions.

In that article, we consider the firm real-time systems where timeliness properties normally guaranteed may admit irregularities in occasional situations mainly caused by prohibitive execution times. A framework should allow a real-time system to gracefully adapt to these exceptions that manifest as deadline missing. The approach presented in this article is a new task model, namely BGW (Black Grey White), developed as a means to provide flexibility in scheduling time-constrained tasks when the processor is overloaded [3].

Scheduling in under-loaded real-time systems

A number of authors have studied the problem of devising algorithms for scheduling time critical jobs on a single processor computing system with no energy consideration. The most popular online scheduling algorithm was introduced by Liu and Layland [4]. According to Earliest Deadline First (EDF) which is preemptive and dynamic priority driven, a ready job with the earliest deadline is executed first. Dertouzos [5] proved that EDF is optimal among all scheduling algorithms on a uniprocessor machine. Consequently, if a set of jobs cannot be scheduled by EDF, then this set cannot be scheduled by any other algorithm. Liu and Layland stated a very simple necessary and sufficient schedulability condition for EDF under the assumption that jobs are the instances of periodic tasks with relative deadlines equal to periods. The EDF strategy is consequently a very desirable approach for scheduling independent jobs preemptively when there is no energy limitation and no processing overload. A survey on EDF scheduling can be found in the work of Buttazzo [6].

Overload management

When the timing constraints cannot be met with computation-quality, one way of maintaining an acceptable Quality of Service in overloaded conditions is to trade computation-quality for timeliness. The Deadline Mechanism offers such possibility with software redundancy [7,8]. In the Deadline Mechanism, two versions of programs are provided for each task: primary and alternate. The primary version is the normal program that produces good quality results whereas the alternate version produces less precise results with very low processor demand. The Deadline Mechanism is integrated in the BGW mechanism.

Another way to reduce the load in transient overloaded conditions is by discarding some jobs that cannot be completed in time under certain conditions. The effectiveness of the so-called Skip-Over approach has been demonstrated especially in multimedia applications and is integrated in the BGW model. In the Skip-Over model, every periodic task τ_i is characterized by its basic timing parameters such as deadline and period and a skip parameter s_i [9]. This parameter represents tolerance for the periodic task to miss some of its deadlines. The distance between two consecutive skips must be at least s_i periods. When s_i equals to infinity, no skips are allowed and τ_i is called a hard periodic task.

Motivations for the BGW model

The design of the BGW model is based on the following assumptions. First, we do not assume a priori knowledge of effective processing times but worst case processing times needed by the application. It can be observed that the amount of processing time needed has high variations as a result of changes in multimedia contents for example. Second, we assume that the application tolerates occasional deadline violations. The BGW mechanism ensures that the tasks execute timely by enforcing certain timing distance between two consecutive successful executions. Third, the application gracefully adapts to overloads by reducing its processing requirements. There are many applications where tasks have stochastic processing times. Such tasks can undergo changes in period too due to application dependencies which result in processing overload. The BGW model provides a systematic approach to specify QoS (Quality of Service) requirements of firm real-time systems wherein timing requirements can be violated during exceptions up to an acceptable known bound.

***Corresponding author:** Maryline Chetto, IRCCyN (Institut de Recherche en Communications et Cybernétique de Nantes), University of Nantes, Nantes, France, Tel : +33 (0) 6 10 20 35 94; E-mail: maryline.chetto@univ-nantes.fr

Received October 24, 2015; **Accepted** October 30, 2015; **Published** November 30, 2015

Citation: Chetto M (2015) Graceful Overload Management in Firm Real-Time Systems. J Inform Tech Softw Eng 5: e124. doi:10.4172/2165-7866.1000e124

Description of the BGW model

The BGW model integrates two approaches:

- The Deadline Mechanism where each real-time periodic task uses two independent versions for the purpose of meeting timing constraints with variable QoS. The BGW model is similar to the Deadline Mechanism, in that both consider any task to be divided into a primary part and an alternate part. However, the Deadline Mechanism imposes to execute at least one of the two algorithms.

- The Skip-Over model where each real-time periodic task has a skip parameter.

Formally, a BGW periodic task set τ is composed of n tasks where each task τ_i is characterized by five timing parameters: r_i Ca_i, Cp_i, D_i and T_i. Ca_i respectively Cp_i refers to the worst case execution time of the alternate respectively the primary, with $Ca_i \leq Cp_i$. D_i respectively T_i refers to the deadline respectively the period of task τ_i.

Two QoS parameters are further defined: n_i and l_i. At least one primary version over n_i successive requests has to execute timely, as well as at least one alternate version over l_i successive requests. The term "distance" can be defined as a number of requests. The parameter n_i expresses the maximum distance allowed between two consecutive successful executions of the primary version. The parameter l_i is the maximum distance allowed between two consecutive successful executions of a job (whatever primary or alternate).

The motivation for the BGW model is to guarantee the stability of computer-based control systems. As the processor may sometimes be overloaded, this model will permit us to determine online the algorithm (primary or alternate) that has to be executed for every task. Control may be realized through two different algorithms: one which returns very precise results but require high processor utilization (primary executed by a black job or a grey job) and one which returns just acceptable results with very short execution time.

A task set is said BGW-schedulable if at least one schedule exists where one primary version over n_i successive requests and one alternate version over l_i successive requests execute completely and timely. A BGW-schedulability test has been given by Ould Sass et al. [3].

At run time, any job generated by a BGW periodic task has one of the three following colors:

- *Black* if the job has to imperatively produce the most precise result with the primary version,

- *Grey* if the job has to produce at least one result, in preference by the primary version,

- *White* if the job may be dropped i.e. the job has no execution requirement even if it is preferable to execute one of the two versions.

Quality of Service of BGW task sets

By definition of the BGW model, the scheduler should execute the primary version of each Black job timely and execute either the primary version or the alternate version of each Grey job timely. As a consequence, at a given time instant for a given task, the scheduler has to execute a job which may be either a black job, or a grey job or it may execute no job at all. The choice depends first on the algorithms (primary or alternate) that have been executed in the past and second on the parameters n_i and l_i of task τ_i. At least one primary version over n_i successive jobs has to be executed timely, and one alternate version over l_i successive jobs has to be executed timely. Moreover, the scheduler should maximize the number of successful primary executions and minimize the number of unsuccessful jobs i.e., jobs which are either discarded or not completed before deadline.

Consequently, the resulting Quality of Service of the system under the BGW model can be measured by the following metrics:

- Number of primary versions which are executed timely over the total number of jobs (NPJ).

- Number of jobs which are executed timely (either by primaries or alternates) over the total number of jobs (NJJ).

Scheduling tasks in overloaded conditions implies to discard jobs that cannot complete before deadline. Consequently, as processor time can be wasted, the cost of different scheduling strategies should be measured by the wasted time ratio (WTR) i.e., the percentage of time used by the processor for producing no result or useless results.

Schedulers for BGW task sets

Two distinct scheduling frameworks may be implemented for the Deadline mechanism. Firstly, according to the First Chance (FC) technique the alternate version of any job executes completely first before the primary version of the same job starts execution. If the primary version finishes before deadline, its results are used in preference to those of the alternate. Secondly, the Last Chance (LC) technique attempts to execute first the primary version. Nevertheless sufficient processing time intervals have to be reserved to guarantee feasible execution of the alternate version if the primary fails. Consequently, success of any primary leads to discard the corresponding alternate and recover processing time since the result of the alternate becomes no longer necessary. In this strategy, the scheduler has to suspend any running primary whenever an alternate requires to be executed so as to meet its deadline.

A scheduling framework for the BGW model is specified by a combination of:

- a scheduling rule for the primaries,

- a scheduling rule for the alternates,

- a scheduling rule for white jobs

- and a hierarchy between the three schedulers.

A job can be in the first list only if it is a black job, in the two first lists if it a grey job and in the third list if it is a white job. Moreover, this framework is hierarchical in that sense that the first two schedulers must be organized under either the First Chance approach or the Last Chance approach. Finally, the list of white jobs has to be served when no jobs are present in the first two lists i.e. as a background scheduler.

Performance Evaluation

Several combinations were studied in simulations. We have implemented the following scheduling strategies: EDF, LCJ and LCP. Under EDF, every task respects the classical Liu and Layland model without QoS parameters. Under LCP, primary versions of grey jobs and primary versions of white jobs are executed as soon as possible. In contrast, primary versions of black jobs and alternate versions of grey jobs are scheduled as late as possible using the Last Chance technique. Under LCJ, primary versions of black jobs and alternate versions of grey jobs are scheduled as soon as possible and alternate versions of white jobs are executed in the remaining idle times. The simulations show that LCP outperforms all other schedulers regarding the metric NPP and LCJ outperforms all other schedulers regarding the metric NPJ. The ratio of deadline misses for the EDF scheduler is never higher

than the ratios for LCP and LCJ. This outlines why both the BGW model and specific scheduling strategies improve significantly the resulting Quality of Service under transient processor overload.

Conclusion

Many applications with content-dependent execution times such as video-processing tolerate bounded non-satisfaction of their timeliness properties. The computing system considered here uses a single-processor machine to run multiple software jobs issued from periodic tasks. We described a new approach for modeling Quality of Service requirements of periodic task systems which may be the object of both transient faults and processor overloads. Several scheduling schemes for the so-called BGW-task model have been implemented and compared. The experiments brought to light the usefulness of this new task model for managing overload in firm real-time systems in comparison to the classical task model.

References

1. Liu JWS (2000) Real-Time Systems. Prentice Hall.
2. Chetto M (2014) Real-time Systems Scheduling. Wiley-ISTE.
3. Ould Sass M, Chetto M, Queudet A (2013) The BGW model for QoS aware scheduling of real-time embedded systems. Proceedings of the 11th ACM international symposium on Mobility management and wireless access: 93-100.
4. Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard real time environment. Journal of the Association for Computing Machinery 20: 46-61.
5. Dertouzos ML (1974) Control Robotics: The Procedural Control of Physical Processes. Proc. of Int. Federation for Information Processing Congress.
6. Buttazzo GC (1997) Hard Real-Time Computing Systems. Kluwer academic.
7. Liestman AL, Campbell RH (1986) A Fault-Tolerant Scheduling Problem. IEEE Transactions on Software Engineering 12: 1089-1095.
8. Chetto H, Chetto M (1989) Some Results of the Earliest Deadline Scheduling Algorithm. IEEE Transactions on Software Engineering 15: 1261-1270.
9. Koren G, Shasha D (1995) Skip-over algorithms and complexity for overloaded systems that allow skips. Proceedings of the 16th IEEE Real-Time Systems Symposium (RTSS'95), Pisa, Italy: 110-117.