An Autonomic Approach for Dynamic Reconfiguration on Real-time Bandwidth-reservation Schedulers

Marco A. C. Simões[1,2] **, George Lima** [1] **, Raimundo Macêdo**[1]

[1]Distributed Systems Laboratory (LaSiD)
Federal University of Bahia (UFBA) – Salvador – BA – Brazil

msimoes@uneb.br,{gmlima, macedo}@ufba.br

[2]Computer Architecture and Operating Systems Group (ACSO)
Bahia State University (UNEB) – Salvador – BA – Brazil

msimoes@uneb.br

Abstract. *This paper presents a novel system model for autonomic and dynamic reconfiguration of real-time systems. The main goal of this model is to provide means for general autonomic reconfiguration, dynamically meeting new requirements from the underlying computer system or from applications, such as creation of new tasks, operation mode change, etc. In this paper we present some experimental data from an early implementation of the proposed model that suggest that our approach can indeed comply with the autonomic reconfiguration challenges we are addressing.*

1. Introduction

The application spectrum of real-time systems has broadened considerably in the past few years. Such systems, once characterized by having simple periodic predictable behavior [Liu and Layland 1973], have become much more complex. Modern hardware and software architectures and/or the support to open environments, for example, make it difficult or even impossible to estimate task worst-case execution times accurately. Furthermore, the environment within which real-time systems are being applied may be highly dynamic. A usual way of dealing with this problem is by providing *temporal isolation* in the system. Indeed, if certain task runs more than what was expected, the effect of this timing fault should not propagate to other application tasks. An usual approach to deal with this problem is by means of bandwidth reservation schedulers. Constant Bandwidth Server (CBS) is one of these approaches that has received special attention recently [Abeni and Buttazzo 1998, Zhou et al. 2006].

Each CB server S_i in the system is defined by a tuple $S_i = (Q_i, T_i)$, where Q_i is the maximum processor time reserved to S_i per T_i time units. In other words, tasks assigned to S_i are guaranteed to have $u_i = Q_i/T_i$ of processor bandwidth. Usually, the parameters (Q_i, T_i) are static and defined off-line. Nonetheless, for certain kind of modern applications, the system should also provide support for *dynamic reconfiguration* of server parameters. Indeed, modern systems may be structured so that their tasks have one or more modes of operation. For example, a control task can execute one among several control functions and/or with one among several sampling periods previously set. In other words, a task that has more than one operation mode may have alternative pieces of code to execute or alternative release periods. Assuming that each operation mode

gives a different benefit for the system, the problem is then to select, at runtime, the set of task modes that maximizes the system benefit subject to the system schedulability.

Dynamic reconfiguring system parameters has been previously addressed by other researches [Beccari et al. 2005, Buttazzo and Abeni 2002, Lima et al. 2008, Rajkumar et al. 1997]. In particular, dynamic adjustment of bandwidth reservation has also been considered. Some approaches are based on applying control theory to tune the server parameters [Lu et al. 2002, Zhou et al. 2006] while others use optimization techniques [de Oliveira et al. 2008, de Oliveira et al. 2009]. In this paper we have a similar goal but we use a different technique. We consider the use of Genetic Algorithms to carry out on-line bandwidth adjustments, similarly to what we have recently described [Simões et al. 2008]. Here we extend this model to make it suitable to autonomic reconfiguration.

In this paper we deal with dynamic reconfiguration in a new and challenging way, as will be seen in Section 2. We present a novel system model that is able to dynamically reconfigure the servers due to applications mode change or QoS monitoring. Some initial experiments are presented in Section 3. Final comments on this work and on the challenges it brings about are given in Section 4.

2. System Model

We consider $T = \{\tau_1, \tau_2, \ldots, \tau_N\}$ as a set of N independent real-time tasks to be executed on uniprocessors. We assume that T is partitioned in two subsets T_{NR} and T_R with $T = T_{NR} \cup T_R$ and $T_{NR} \cap T_R = \emptyset$. T_{NR} is the set of non-reconfigurable tasks, which have a unique mode of operation during the system life time. In other words, they are not subject to any reconfiguration procedure and correspond to usual hard real-time tasks. T_R is the set of n reconfigurable tasks. They can be soft or hard real-time tasks with uncertain or imprecise execution parameters or with different modes of operation. For example, a video-encoding task can operate in distinct modes depending on the required video resolution. These modes may require different processing resources. Furthermore, within each mode, the necessary processing resources may vary and cannot be predicted beforehand.

Due to its time uncertainty, each task in T_R is assigned to one Constant Bandwidth Server (CBS) [Abeni and Buttazzo 1998] from the set $\mathcal{S} = \{S_1, S_2, ..., S_n\}$, which are scheduled by EDF [Liu and Layland 1973]. This means that *temporal isolation* is provided within the system. The reconfiguration procedure tunes each server parameter so that the system can be adjusted at run time according to the current demand. In other words, the reconfigurator may choose different values of Q_i or T_i for each server $S_i \in \mathcal{S}$ so that either QoS or mode change requests are observed.

Fig. 1 illustrates the proposed model. There are four main components: *scheduler*, *system monitor*, *adaptation manager* and *reconfigurator*. The scheduler utilizes a bandwidth reservation mechanism to provide temporal isolation, as described above. The system monitor is periodically invoked to collect statistics about system execution regarding previously defined QoS metrics. For example, the monitor can detect some overload scenarios evaluating deadline miss ratio, tardiness, lateness, etc. When the monitor identifies an indication of system changing, in particular in QoS of tasks in T_R, it can invoke a system call *reconfig*, activating the adaptation manager. Then, reconfiguration is carried

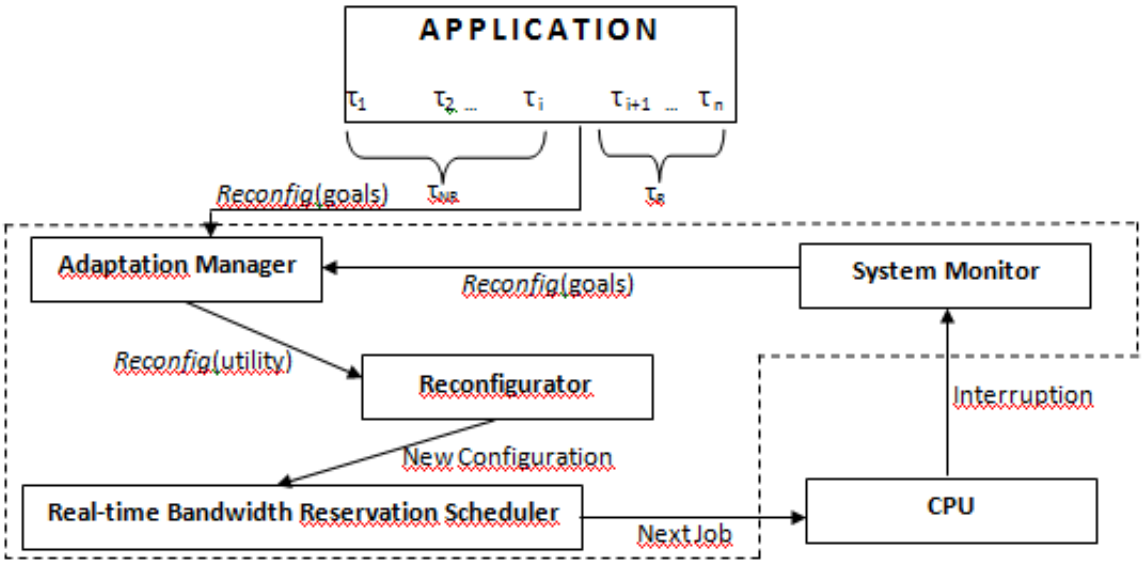

Figure 1. Dynamic Reconfiguration Model

out by the reconfigurator. The reconfiguration goals are informed by the adaptation manager. These goals can be associated to minimizing deadline miss ratio, for instance. The adaptation manager also acts when application requirements are dynamically modified, for instance, when a new task is created or when an existing task is removed from the system.

Suppose reconfiguration is required to select modes of operation for each task in T_R. Consider that each task $\tau_i \in T_R$ or its server S_i has $\kappa(i) \geq 1$ modes. Its set of modes is denoted as $K_i = \{1, \ldots, \kappa(i)\}$. The k-th operation mode of S_i is defined as the tuple (Q_{ik}, T_{ik}), where Q_{ik} represents the server mode capacity and T_{ik} is its period. Thus, each server S_i operating in mode k has a guaranteed processor utilization of $U_{ik} = \frac{Q_{ik}}{T_{ik}}$. Assuming that if task τ_i operates in mode k gives a benefit for the system A_{ik}, the goal could be to maximize the overall benefit. In this case, a possible formulation for the reconfiguration problem can be expressed as follows.

$$R : f = Maximize \sum_{S_i \in S} \sum_{k \in K_i} A_{ik} x_{ik} \tag{1a}$$

$$\sum_{S_i \in S} \sum_{k \in K_i} U_{ik} x_{ik} \leq 1 \tag{1b}$$

$$U_{ik} = \frac{Q_{ik}}{T_{ik}} \tag{1c}$$

$$\sum_{k \in K_i} x_{ik} = 1,\ S_i \in S \tag{1d}$$

$$x_{ik} \in \{0, 1\},\ S_i \in S,\ k \in K_i \tag{1e}$$

Equation (1a) defines the objective function. Variable x_{ik}, defined by equation (1e), represents the choice of one of the $k \in K_i$ server operation modes of S_i. Only

one configuration must be selected by `reconfig`, which is ensured by equation (1d). Restriction (1b) guarantees the schedulability of S according to the EDF policy.

Other formulations for the reconfigurator are possible depending on the system needs. For example, the objective for certain tasks could be to minimize their lateness while for others what matters is deadline miss ratio. Indeed, one can formulate multi-objective functions to incorporate such required behaviors. In this paper we propose to use Genetic Algorithm (GA) so that a generic optimization framework can be implemented. By doing so we expect to provide support for a large spectrum of formulated utility functions. Furthermore, GA procedures can be used for gradually adapting the system to environment changes since the optimization can start from the previous solution. Obviously that there are research challenges to be addressed. For example, it is known that GA may converge slower than other optimization techniques [de Oliveira et al. 2009]. This may require that higher processing resource is reserved for reconfiguration. In the next section we illustrate the performance of the proposed model using formulation (1a)-(1e).

3. Experimental Results

In this section we illustrate the proposed architecture by running a simple experiment. We simulated randomly generated outputs from the adaptation manager, performing several *reconfig* calls to the reconfigurator using different utility functions. The GA-based reconfigurator used here is similar to the one described in [Simões et al. 2008]. Unlike this former work, though, we are considering in this experiment that benefit values change during the simulation.

3.1. Experiment Setup

Initially, we considered a system composed of $n = 10$ CBS servers but the number of new tasks increased during the simulation up to $n = 15$. To each server was assigned a task τ_i with $\kappa(i) \geq 1$ operation modes. For this experiment, we initialized the system with $\kappa(i) \leq 15$ and considered the formulation given by (1a)-(1e). The reconfigurator ran on a server S_{rec} with $Q_{rec} = 20$ and $T_{rec} = 100$. The population size used in GA was set to $5n$ and each individual was codded in binary. The operations for selection, crossover, mutation and migration were configured such as in [Simões et al. 2008].

The reconfigurations are simulated by calling the function `reconfig`. Before each call the values of the modes (Q_{ik}, T_{ik}) and their associated benefit values $A_{ik} \in [0, 1]$ were randomly generated for all tasks τ_i, $i = 1, \ldots, n$ using uniform distributions. The values of Q_{ik} were generated in $[10\,ms, 100\,ms]$ and processing utilization U_{ik} were in $(0, 0.2]$. Then, the corresponding values of T_{ik} were set to $T_{ik} = \frac{Q_{ik}}{U_{ik}}$. When calling `reconfig`, it is indicated which mode of operation is desired for each task by `reconfig`$(k_1, k_2, \ldots, k_n)$, where k_i, $1 \leq k_i \leq \kappa(i)$, is the requested operating mode for task τ_i. By modeling the reconfigurations in this way we are considering arbitrary benefit values for a discrete optimization formulation. Although these mode change requests will hardly be found in practice, they provide a good assessment mechanism. Indeed, in this experiment the benefit a task has in one mode for a given request is not related to the values regarding the other calls. It is worth mentioning that the evaluations previously carried out in other work [de Oliveira et al. 2009, Simões et al. 2008] have taken simpler set-ups.

3.2. Results

Even considering the above described simulation set-up, the results indicate a similar performance when compared to the former work [Simões et al. 2008], for which the same mode change requests were performed with fixed utility functions.

A metric, called Average Performance Ratio (APR), was defined as $APR = \frac{1}{m}\sum_{j=1}^{m}\frac{V_j}{OPT_j}$, where m is the number of reconfiguration requests. APR indicates how close to the optimum the solutions provided by the reconfigurator are. APR was computed considering solutions obtained when the GA procedure stabilizes. The value of m was set to 100 and the interval between consecutive calls was $1000ms$. The optimum value OPT_j for each request j was obtained through a dynamic programming procedure (not shown here).

The proposed approach reaches $APR = 78\%$ with an average convergence time of $592\,ms$. Fig. 2 illustrates the obtained distributions of both APR and the convergence time of the reconfigurator. On average, the reconfigurator converged with 52.59 iterations of GA consuming 13.51 activations of S_{rec}. These results indicate that GA is a feasible solution to provide an autonomic reconfiguration mechanism.

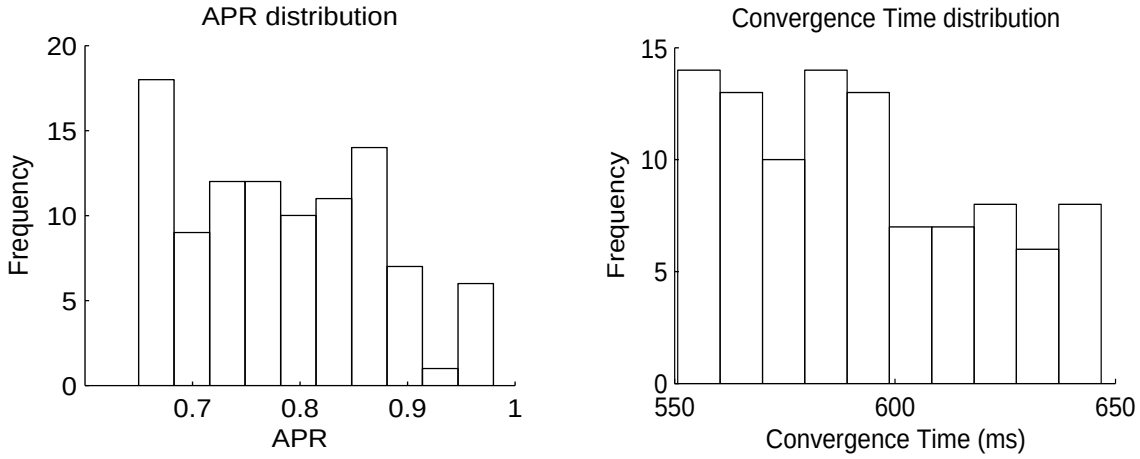

Figure 2. APR and Convergence Time results

4. Conclusions and Future Work

In this paper we presented a novel system model for autonomic and dynamic reconfiguration of real-time systems. The main goal of this model is to provide means for general autonomic reconfiguration, dynamically adjusting the underlying computer system as well as applications new to new requirements. Such functionality is required to support, for instance, emerging self-manageable soft real-time systems [Andrade and de Araújo Macêdo 2009]. In order to attain the required reconfiguration capability, we made use of genetic algorithms that can handle dynamic changes of utility functions, representing new system or application requirements.

We also conducted an initial evaluation of a partial implementation of the model that suggests that our approach can indeed comply with the autonomic reconfiguration challenges we are addressing.

In future work we will enhance the *reconfigurator* to get better values for APR and convergence time, applying it to more complex and multiobjective functions, leading us to demonstrate the real genericity of our solution. It is also left for future work the whole implementation of the model, which includes the development of proper system monitors and adaptation managers. The application of the whole model to synthetic and real applications will complete the validation of this proposal.

References

Abeni, L. and Buttazzo, G. (1998). Integrating multimedia applications in hard real-time systems. *Real Time Systems Symposium(RTSS), The 19th IEEE*, pages 4–13.

Andrade, S. S. and de Araújo Macêdo, R. J. (2009). A non-intrusive component-based approach for deploying unanticipated self-management behaviour. In *Proceedings of IEEE ICSE 2009 Workshop Software Engineering for Adaptive and Self-Managing Systems*.

Beccari, G., Caselli, S., and Zanichelli, F. (2005). A technique for adaptive scheduling of soft real-time tasks. *Real-Time Systems*, 30(3):187–215.

Buttazzo, G. and Abeni, L. (2002). Adaptive workload management through elastic scheduling. *Real-Time Systems*, 23(1-2):7–24.

de Oliveira, A. B., Camponogara, E., and Lima, G. (2008). Dynamic reconfiguration of constant bandwidth servers. In *Proc. of the 10th Brazilian Workshop on Real-Time and Embedded Systems (WTR 08)*, pages 37–44. Brazilian Computer Society.

de Oliveira, A. B., Camponogara, E., and Lima, G. (2009). Dynamic reconfiguration in reservation-based scheduling: An optimization approach. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE. to be published.

Lima, G., Camponogara, E., and Sokolonski, A. C. (2008). Dynamic reconfiguration for adaptive multiversion real-time systems. In *Proc. of The 20th IEEE euromicro Conference on Real-Time Systems (ECRTS 08) - to appear*, pages 1–10, Prague, Czech Republic, 2008.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61.

Lu, C., Stankovic, J. A., Son, S. H., and Tao, G. (2002). Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real-Time Syst.*, 23(1-2):85–126.

Rajkumar, R., Lee, C., Lehoczky, J., and Siewiorek, D. (1997). A resource allocation model for qos management. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 298–307.

Simões, M. A. C., Lima, G., and Camponogara, E. (2008). A ga-based approach to dynamic reconfiguration of real-time systems. In *Proceedings of Workshop on Adaptive and Reconfigurable Embedded Systems (APRES '08)*, St. Louis, MO, USA. IEEE Computer Society.

Zhou, P., Xie, J., and Deng, X. (2006). Optimal feedback scheduling of model predictive controllers. *Journal of Control Theory and Applications*, 4(2):175–180.