A bi-constraints adaptation technique for embedded multimedia systems

Mouna Ben Saïd, Nader Ben Amor, M. Abid, F. Ben Taher
CES Laboratory
National Engineering School of Sfax, University of Sfax,
Tunisia
mouna.bensaid2@gmail.com

J. Philippe Diguet
Lab STICC
UBS University
Lorient, France

***Abstract*— The growing number of complex multimedia applications has raised new challenges in the design of embedded mobile multimedia systems. These systems must be able to react to resource limitations and external environment fluctuations to meet applications' timing constraints and provide an acceptable service. In this paper, we present a new adaptation technique enabling a multi-task system to adapt to variations in both processor-load and network bandwidth resources. Adaptations are performed by dynamically reconfiguring the running applications to meet shared resource allocations using a new resource sharing algorithm. Tests were done on the H.264/AVC video compression application.**

***Index Terms*—Adaptation, embedded systems, H.264/AVC, MO-PSO, multimedia application, resource sharing.**

I. INTRODUCTION

The rapid development of multimedia applications has raised challenges in the design of multimedia nomad systems. They must provide a minimal acceptable Quality of Service (QoS) in spite of resource limitations (energy and computing) and external environment fluctuations.

Minimizing complexity, while maximizing QoS, presents an inherent conflict in the design of mobile systems. A high QoS needs a high utilization of system resources like CPU cycles or additional HW resources and leads consequently to high power consumption. A low QoS would utilize few resources and so consume low power, but yield low performance.

Although the requirement of high performance and low energy consumption is challenging, it is becoming achievable with the appearance of adaptable system layers ranging from hardware to application. Indeed, multimedia applications have the ability to gracefully adapt to resource fluctuations while keeping a meaningful user's perceptual quality.

In the present work, we propose a bi-constraints adaptation technique for embedded systems running CPU/bandwidth-intensive multimedia applications. This technique considers adaptation at the application level at different time granularities and under different constraints. It adapts the system to resource constraints and user preferences. It leads a coarse-time granularity adaptation to the changing CPU load (task entry or exit) and manages a fine-time granularity adaptation to external resource related to network bandwidth fluctuation.

We perform these adaptations by dynamically reconfiguring the running applications to scale their QoS levels in order to meet resource allocations. A minimal quality of service defined as a user preference is maintained. We define adaptation aware resource sharing algorithms to update resource allocations among competing tasks at every adaptation event. As a case study, we consider a H.264/AVC video encoder to apply the adaptations.

The paper is organized as follows. Section II gives the major features of our work and compares it with related works. Section III presents the design of the proposed technique. We introduce the problem then describe the architecture and the developed resource sharing algorithm. Section IV shows the implementation and test. Finally, section V concludes the paper.

II. RELATED WORKS

Scaling the application QoS level has an influence on CPU utilization. By selectively reducing the QoS level provided to individual applications of a multi-task system, we can get a reduced system overall complexity and thus a reduced system load [1]. Scaling QoS level can be performed by numerous techniques which have been developed in the literature. This includes algorithms and parameters modification in the application layer, and adapting scheduling policies to resource variation in the OS layer.

Among these techniques, multiple works have shown the effectiveness of application adaptation. They have also shown the importance of integrating network adaptation in multi-task network-constrained systems. Authors in [2] have illustrated that application-aware adaptation offers the most general and effective approach to mobile information access. In [3], authors of the GRACE project have shown the importance of per-application (per-app) adaptation at both application and network layers. They have demonstrated that while running multiple applications on a network bandwidth constrained environment, the lack of network awareness results in very modest benefits from the hierarchical adaptation (global and local adaptation). They have also shown that in such environment, per-app application adaptation yields significant energy benefits over and above global adaptation.

Adaptation based on application QoS-levels variation has been used in the literature, but methods to define these levels are still under-explored. For example, authors in [5] and [6] define hardware and software configurations for a 3D synthesis application. These configurations yield a very limited number

978-1-61284-732-0/11/$26.00 ©2010 IEEE

of tuned parameters (number of triangles of an object and shading algorithm) and are thus easily prepared in a manual way. However, such a method is inappropriate for more complex applications such as video coding which exhibit a very important number of possibly tuned parameters. Thus, we propose a novel method of an automatic generation of QoS-levels for applications with a large set of configuration parameters. It is based on the Multi-Objective Particle Swarm Optimization algorithm. This method enables the generation of non-dominated QoS-levels and thus obtaining a maximized system output quality.

III. THE PROPOSED ADAPTATION TECHNIQUE

A. Global Overview

We consider that our target system is network constrained and has to run multiple concurrent multimedia applications. Such system is sensitive to dynamically varying network bandwidth and system load which affect the perceived quality and consequently the user's satisfaction. Hence, they must provide a minimum of QoS in order to guarantee the continuity of service in the presence of fluctuating environment. We consider developing an adaptation technique which is network-aware and based on per-application adaptation. One of the major features of our adaptive technique is that it manages adaptations to different resources at application level and at different time granularities. Fortunately, the soft real-time nature of many multimedia applications offers more opportunities for QoS trade-off. For example, video codecs such as H.264/AVC are highly configurable. They offer a large number of possibly tuned parameters to encode and decode videos with different compression ratios, different computational complexities, and variable output quality. We propose a global resource allocation manager which enforces a per-application software reconfiguration in such a way that the system overall output quality is maximized under available resource constraints, which are the CPU time and the network bandwidth.

The essence of our adaptation technique is the coordination between resource allocation and application adaptation in response to dynamic resource changes in a multi-task mobile system. The change may be caused by variation in a resource supply due to fluctuant environment or system mobility, or by changed demand for it by concurrent tasks (an application is assumed to be one task).

In this work, we consider two kinds of variations: changes in task number (i.e., task entry or exit) and changes in the network bandwidth availability. These variations trigger adaptations at different time scales. The former handles large system changes at coarse time granularity, and requires reallocating both CPU time and bandwidth budgets among tasks (e.g., in minutes or per-task), while the latter adapts the generated bit rate according to the available bandwidth at fine time granularity (e.g., in tens of milliseconds or per job).

The structure of the proposed technique is depicted in Figure 1.

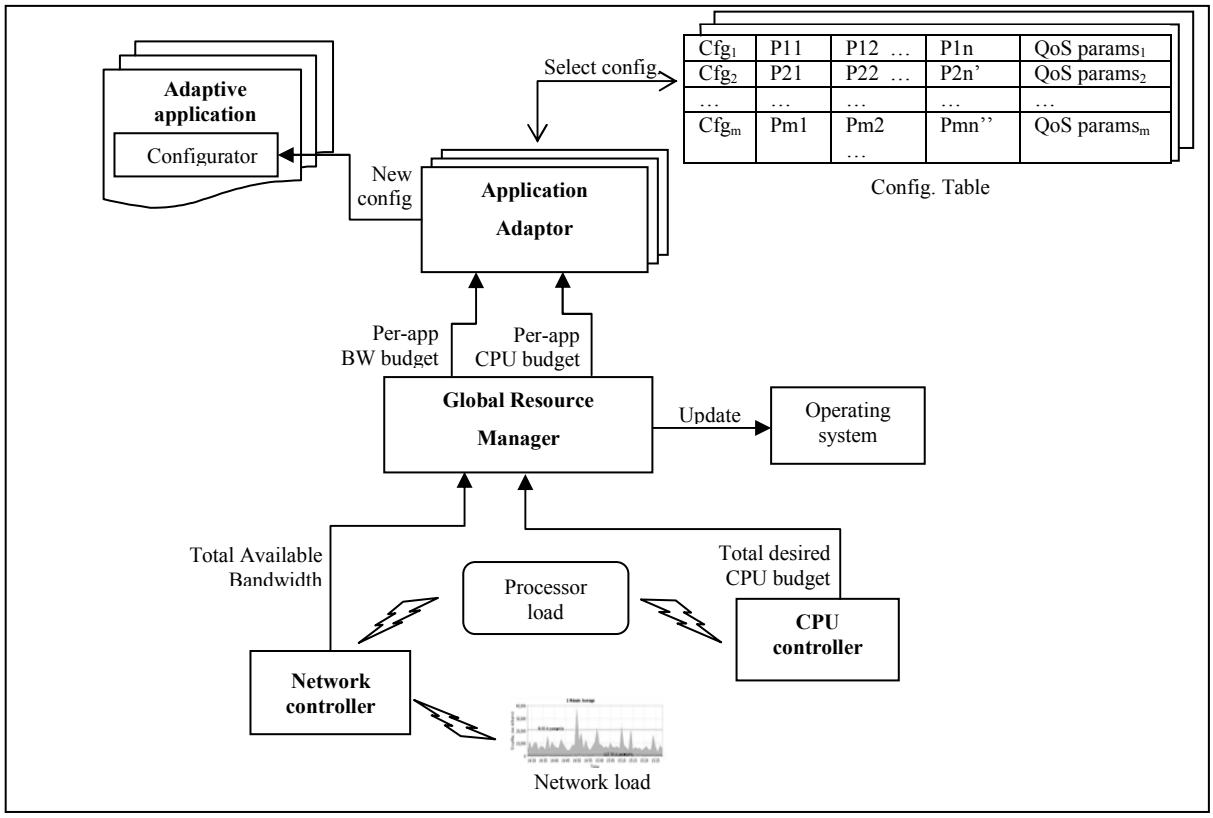

Figure 1. *Structure of the bi-constraints adaptation technique*

Resource monitoring is handled by two modules: i) specific resource controllers which notify, at different time scales, ii) a global resource manager (GRM) of resource variations and claim resource re-allocation. We define a CPU controller which manages the infrequent variation of processor load. It notifies the GRM of variation in total CPU budget demand at every task entry or exit event. We also define a network controller which handles both system load variation and frequent network bandwidth fluctuation. It notifies, as well, the GRM of the total available bandwidth in response to a bandwidth fall or rise event due to change in either the CPU or the network load.

Upon receiving a new resource constraint, the GRM saves this latter, preserves complete budgets demands for non-adaptive applications, and re-allocates budgets for adaptive ones. Regarding the CPU constraint, it performs a schedulability analysis of the running task set. It adjusts adaptive CPU budgets so that it fully exploits, without exceeding, the total CPU time. As for the bandwidth constraint, the GRM adjusts total bandwidth budget to the new available bandwidth by acting on the adaptive bandwidth budgets. We should note that the GRM deals with both over and under resource utilization.

After resource budgets re-allocation, the GRM sends new per-application resource budgets to application adaptors. Each application adaptor maps the received budgets allocation to application-specific configuration.

First, a configuration is selected from a pre-established configuration table that best responds to allocated resource budgets. After that, an application-specific configurator dynamically reconfigures the application with the adaptation choices.

B. Problem formulation

We describe our adaptive system as follows. Let's have a set of n periodic tasks *T1…Tn* running concurrently on the system. Each task *Ti* has *mi* different configurations, that we also call QoS levels, and are defined as $L_{i,1}...L_{i,mi}$. For each configuration *Li,j* we specify the task real-time characteristics. Let Pi,j be its period, and Ci,j be the average CPU processing time per job. For example, a video encoder at a frame rate equals to 25 fps has a period 40 ms for each frame coding job. We specify also an average bit rate Bi,j and QoS value Qi,j. Let Bw be the available network bandwidth and Pbudi the CPU budget required by each task i.

In a case of a mono-task system, we simply need to select the configuration of the running application that best satisfies the CPU time constraint (i.e. Ci,j ≤ Pi,j), and the bandwidth availability (i.e. Bi,j ≤ Bw), while maximizing the individual application QoS. However, in a multi-task system, the purpose of our adaptation model will be the selection of a combination of per-task QoS-levels $L_{1,j}...L_{n,k}$ that maximizes the system overall quality subject to resource constraints. We formulate hereafter this problem as a constrained optimization problem.

We consider a binary variable xij such that:

x_{ij} 1 | The j^{th} configuration of the i^{th} application is selected

0 | else

The adaptation scheme has to select a combination of adaptive applications' QoS-levels to:

$$\text{Maximize } \left(\sum_{i=1}^{n}\sum_{j=1}^{mi} \boldsymbol{Qij} * \boldsymbol{xij}\right) \quad (1)$$

Subject to:

$$\sum_{i=1}^{n}\sum_{j=1}^{mi} \frac{\boldsymbol{Cij}}{\boldsymbol{Pij}} * \boldsymbol{xij} \leq \mathbf{1} \quad (2)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{mi} \boldsymbol{Bij} * \boldsymbol{xij} \leq \boldsymbol{BW} \quad (3)$$

$$\sum_{j=1}^{mi} \boldsymbol{xij} = \mathbf{1} \quad \forall i = 1,2,\ldots,n \quad (4)$$

To solve this problem we allocate for each application i a budget of available CPU time, CPUbgt_i, and available network bandwidth, BWbgt_i. The sum of allocated budgets has to comply with the CPU and BW constraints (2)(3). Each application has a specific adaptor which is responsible for selecting the best configuration that maximizes the quality and meets the allocated budgets.

To ensure fair distribution of the available resources (bandwith and CPU time) to the various tasks, a resource sharing algorithm is used.

C. Configuration table set up

The generation of the configuration table is based on a multi-objective optimization algorithm which:

- Maximise (Qi,j)
- Minimise (Ci,j)
- Minimise (Bi,j)

In this paper, we adopt the MO-PSO algorithm.

D. bi-constraints adaptation

1) CPU adaptation: the role of our CPU adaptation is to deal with the real-time constraint when the system runs a variable number of tasks. This involves a dynamic change of the alloctaed time for each task. The basic idea of the CPU adaptation is to adjust the CPU allocation for each task based on a schedulability analysis. In this paper, we use the Earliest Deadline First (EDF) algorithm [7].

2) Bandwidth adaptation: Our bandwidth adaptation is dedicated essentially to deal with the problem of video quality control of streaming video over drastically changing networks. In fact, the frame rate of the video streaming is affected by the network bandwidth availability. Our goal is to maximize the quality of the delivered signal, taking into account the available bandwidth. Bandwidth adaptation is triggered when detecting either a shortage or an under-utilization of the available bandwidth.

E. Resource sharing algorithm

Resource sharing algorithms are based on the notion of Fairness. Given the resource consumption rates demanded by a flow, fairness specifies a particular rate of resource consumption for each flow. Multiple formal notions of fairness have been proposed in the literature to precisely define what is fair in resource allocation amongst competing flows. These include, among others, max-min fairness and weighted fairness [9]. A max-min fair share algorithm is based on maintaining high budget for the sources with smallest demands. These sharing algorithms deal with competing entities with no regard to their specific constraints. They don't take into consideration

the tolerance of the entities to resource adaptation for example. For this purpose, we were brought to define a custom resource sharing algorithm that supports our adaptation mechanism.

This new algorithm firstly makes a difference between tasks by prioritizing non-adaptive tasks over adaptive ones. It divides the shared resource so that non-adaptive tasks are allocated their full demands. Secondly, it guarantees that adaptive tasks are allocated the minimum necessary resource budget enabling them to achieve their minimum acceptable quality. The remaining amount of resource is divided among all adaptive tasks proportionally to their weights.

Therefore, the fairness goals of our sharing algorithm are as follows:

- Allocate resources fairly based on tasks weights
- Allocate full requirements of non-adaptive tasks
- Guarantee a minimum of quality for every task
- Enable full utilization of resource if no contenders

The GRM performs the resource sharing at every new task entry, task exit, or bandwidth variation event. Due to the paper size, we only give the algorithm triggered by new task entry event. We note that these algorithms are performed with coordination between the resource controllers and the GRM. For generality purpose, we use the letter "R" to designate the resource. The algorithms fit both CPU and bandwidth resources. Fig 2 shows the developed algorithm.

When a new task enters the system, it has to define a maximum, *maxR*, and a minimum, *minR*, of resource demands. R is the amount of available resource used to calculate task budgets. For CPU time resource, R represents the period of the task. For network bandwidth, it designates the total bandwidth available for the system.

Task Entry (R, maxR, minR)

Begin

MinR_Bgt $\leftarrow \frac{minR}{R}$

totMinR_Bgt ← *totMinR_Bgt* + *minR*

If (*totMinR_Bgt* > 1) **Then**

Reject new task

Exit

EndIf

R_bgt $\leftarrow \frac{maxR}{R}$

totR_Bgt ← totR_Bgt + R_bgt

If (task is *adaptive*) **Then**

AdaptiveR_Bgt ← *R_bgt – minR_Bgt*

totAdaptiveR_Bgt ← *totAdaptiveR_Bgt* + *adaptiveR_Bgt*

Else

AdaptiveR_Bgt ← *0*

EndIf

For each adaptive task

R_bgt ← R_*bgt* - $\frac{adaptiveR\_Bgt}{totAdaptiveR\_Bgt}$ * (*totR_Bgt* - 1)

AdaptiveR_Bgt ← *R_bgt – minR_Bgt*

maxR ← R_*bgt* * *R*

EndFor

totR_Bgt ← 1

totAdaptiveR_Bgt ← 1 – *totMinR_Bgt*

End

Figure 2. *The resource sharing algorithm*

We define for each task a set of resource budgets: a total allocated budget, *R_Bgt*, a minimum resource budget, *minR_Bgt*, and an adaptive resource budget, *adaptiveR_Bgt* (if the task is adaptive). Initialization and relation between these budgets is given by the equations in Figure 2.

The entering task undergoes an admission test using *totMinR_Bgt* which is the sum of minimum budgets of all running tasks. It is based on a feasibility test of the worst case trajectory of the new task set, i.e. when all tasks are assigned their minimum requested budgets. If the worst case trajectory is not feasible (*totMinR_Bgt* > 1) then the new task is rejected. Otherwise, the new task entry is performed successfully and we go through all tasks to re-allocate budgets. If the task is non-adaptive, its *R_bgt* is not altered (i.e. fully allocated). If it is adaptive, it is allocated a *R_bgt* proportional to its defined weight. At the end, the total allocated resource budget and the total adaptive resource budget are updated.

IV. IMPLEMENTATION AND TEST

Our technique is dedicated to embedded systems running concurrent multimedia applications, such as audio and video codecs that are long-lived (e.g., lasting for minutes or hours) and CPU/bandwidth intensive. In this paper, we present a PC-based platform. We are planning to extend it for FPGA platforms. We use as a case study the H.264/AVC encoder JM16.2 [10,11,12].

A. Configuration table set up

Our adaptation framework needs a configuration table per application, whose records represent the generated application QoS levels. It is based on an offline characterization using two main steps:

1) Design of an application configuration: We characterize a QoS level by a set of n configuration parameters *{p1, p2, ...pn}* (i.e. whose values can be changed dynamically), and p QoS parameters *{QoS1,...QoSp}* which are application dependent. A QoS level is then a multidimensional variable of *(n + p)*-tuples representing a combination of values of the n configuration parameters.

Multiple possibilities exist when selecting the configuration parameters for the H.264/AVC encoder due to its high parameterization and flexibility. We consider four configuration parameters which are *IntraPeriod*, *QPISlice*, *SearchMode*, and Inter Mode Prediction Control parameters (such as *PSliceSearch8x4*, *PSliceSearch4x8*, and *PSliceSearch4x4*)[11]. We use as constraints the PSNR (QOS parameter), the average encoding time/frame *Tex* and the bit rate *Br*.

2) Automatic configuration generation: For the purpose of automation of the configurations generation, we use the Multi-Objective-Particle Swarm Optimization (MO-PSO) algorithm [13]. The selected parameters are introduced to the MO-PSO algorithm to generate QoS-scalable parameter sets which range from < highest quality/highest complexity > to < lowest quality/lowest complexity>. In our case (multi-objective optimization), there is a set of different solutions, the so-called Pareto optimal set [14] i.e. the set of application configurations that generate the best QoS for a given processing time and bit rate.

The main steps needed to link the JM encoder to the MO-PSO algorithm are as follow:

- We give the definition of particles in initialize_pop() function. We define for each particle the composing variables (configuration parameters) and their value ranges.
- We define three objective functions, H264_Q(), H264_BR() and H264_Tex(), which return respectively the QoS, Bit rate and processing time values of a given particle.
- Evaluate particles in population.

3) MO-PSO results: there are no explicit rules to tune the parameters of the MO-PSO (number of particles, number of generations, and archive size). The best parameters tuning depends on the application considered for optimization, and are determined after multiple tests. For simplification reasons, we consider that only one type of video runs on the system. The videos we consider for test are speaking head videos such as "foreman.yuv" and "news.yuv". Figure 3 illustrates the pareto front generated using an MO-PSO with 50 particles, 50 generations, and 50 archive solutions. We obtain a good result that is enough convenient with our adaptation needs: the solutions are enough spread out so that they cover different QoS level ranges and different scenarios of resources variation (only CPU constraint, only bandwidth constraint, or both CPU and bandwidth constraints).

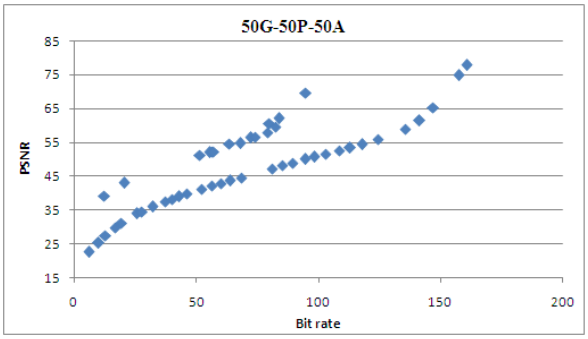

Figure 3. MO-PSO pareto front for 50G-50P-50A, bit rate vs. PSNR

B. The online adaptation

The adaptation API[1] is composed of two parts. The first part is application independent which manages resource variation. The second part is application specific part that is hooked into the application to enable its reconfiguration. We do the necessary code modification of the original JM encoder to enable its dynamic reconfiguration (parameter change while coding) [15].

C. Tests

We consider a set of 4 tasks: a video encoder task, *H.264_enc*, and 3 other multimedia tasks *MM2*, *MM3*, and *MM4*, with different resource variations. The MM2 task is non-adaptive and the three others are adaptive ones. TABLE I presents the list of tasks that enter (TE) or exit (TX) the system with the corresponding moment of occurrence (frame number).

TABLE I. TEST SCENARIO

Nb frame	Event	Texe (ms)	MinTexe(ms)	Period(ms)	BR (kb/s)	MinBR (kb/s)
0	TE H264	500	100	500	1400	28
32	TE MM2	300	100	300	800	300
60	TE MM3	30	30	100	50	50
120	TE MM4	100	80	300	500	100
140	TX MM2	-	-	-	-	-
220	TX MM3	-	-	-	-	-

1) Reconfiguration behavior: the curves depicted in Fig.4 summarize the network bandwidth variation and mention the occurrence of the H.264 encoder reconfiguration. The encoder first reconfigures when new entering tasks require important CPU and bandwidth resources (TE MM2 and TE MM3). It also selects a better configuration when its CPU and bandwidth budgets greatly increase thanks to the MM2 task exit at frame number 140. We remark that the application doesn't reconfigure at every bandwidth variation and thus is able to achieve a stable status during different intervals. This is thanks to the interval tolerance mechanism that has been implemented in the application adaptor. The thick line on the top of the figure designates the requested bandwidth value of a non adaptive video encoder during the test scenario.

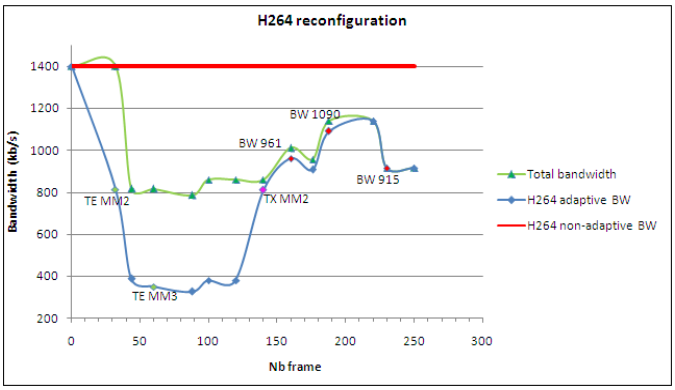

Figure 4. *Reconfiguration behavior of the encoder following resources variation*

2) Impact of adaptation on video quality: we illustrate in Fig. 5 the major quality variation of the encoded video. We notice that the encoder manages to scale gracefully its video quality according to the system resources availability in order to maintain the same video frame rate. The video quality decreases whenever the number of tasks increases (images (b) and (d)) or a bandwidth shortening occurs (image (c)). The image (f) corresponds to a decision of reconfiguration to upgrade the video quality when an increase of the system available resources occurs (task MM2 exits).

[1] *Application Programming Interface*

Figure 5. Impact of adaptation on video quality

We calculated the average overheads of the different adaptation modules using a HP laptop with a 1.78GHz processor. They are depicted in *TABLE II*. These overheads are considered negligible compared to CPU intensive multimedia applications (up to 500ms of execution time/frame in our case) which, indeed, tolerate constraints exceeding to some extent.

TABLE II. *ADAPTATION OVERHEAD*

Adaptation module	Overhead (ms)
Adjust Bandwidth budgets	0.07
Adjust CPU budgets	0.07
Reconfigure H.264 application	6 ms

The overhead of the resource adjustment modules is proportional to the number of running applications but the complexity of the algorithm is linear. The reconfiguration module is application-specific. Thus, its overhead depends on the complexity of the application reconfiguration.

V. CONCLUSION

In this work, we dealt with the set up, on a multi-task system, of an adaptation technique applied to the application level (software adaptation). We considered adapting the system to changes in both internal and external resources, namely the available network bandwidth and the CPU load. Adaptation is performed by scaling the application QoS level according to resources availability. We adjust the configuration of adaptive running applications to meet resources constraints while maximizing the output quality. We defined two modules to handle resource monitoring: specific resource controllers which monitor resources changes at different time granularities, and a global resource manager responsible for shared resource reallocation among competing tasks. We developed our own resource sharing algorithm. We applied our adaptation on the H.264/AVC encoder.

The adaptation technique was tested using an automatically generated configuration database for the H.264/AVC standard. Results show that thanks to the adaptation mechanism, the system is able to scale gracefully its output quality according to the environment fluctuation while guaranteeing the continuity of the service.

We plan in our ongoing work to map the application on an RTOS and run it on the target system to test its auto-adaptation and concurrency management capabilities on an FPGA platform.

REFERENCES

[1] P. Pillai, H. Huang, and K.G. Shin, *"Energy-Aware Quality of Service Adaptation,"* Technical Report CSE-TR-479-03, Univ. of Michigan, 2003

[2] Noble, B. D., Satyanarayanan, M., D.Narayanan, J.E.Tilton, and Flinn, J. *"Agile application-aware adaptation for mobility"*. In Proc. of 16th ACM Symposium on OS and Principles, France, Oct. 1997

[3] Vibhore Vardhan, Daniel G. Sachs, Wanghong Yuan, Albert F. Harris, Sarita V. Adve, Douglas L. Jones, Robin H. Kravets, and Klara Nahrstedt, *"Integrating Fine-Grained Application Adaptation with Global Adaptation for Saving Energy,"* Int. J. Embedded Systems, 2007

[4] S. Saponara, M. Casula, F. Rovati, D. Alfonso, Member, IEEE and L. Fanucci, Member, IEEE, *"Dynamic Control of Motion Estimation Search Parameters for Low Complex H.264 Video Coding,"* IEEE Transactions on Consumer Electronics, Vol. 52, No. 1, Feb. 2006

[5] N. BEN AMOR, *" Design approach for embedded vision processor,"* Thèse de doctorat, ENIS, Dec. 12, 2005

[6] K. Loukil, N. Ben Amor, M. Abid, *"Self adaptive reconfigurable system based on middleware cross layer adaptation model,"* SSD, Djerba, Tunisia, Apr. 2009

[7] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, Jan. 1973

[8] Yunkai Zhou and Harish Sethu, *"On Achieving Fairness in the Joint Allocation of Processing and Bandwidth Resources,"* Quality of Service — IWQoS 2003 book, Springer-Verlag Berlin Heidelberg, Vol. 2707/2003, pp. 97–114, 2003.

[9] Ivan Marsic, "*Computer and communication network*," Adapted from: S.Keshav, *An Engineering Approach to Computer Networking*, Addison-Wesley, Book, 1997, pp 215-217, 1998

[10] http://iphome.hhi.de/suehring/tml, The JM software web site

[11] Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, *"H.264/14496-10 AVC reference software manual,"* document JVT-AEO10, Jan. 2009

[12] G. J. Sullivan, and Th. Wiegand, *"Video Compression—From Concepts to the H.264/AVC Standard,"* proceedings of the IEEE, vol. 93, No. 1, January 2005

[13] Margarita Reyes-Sierra and Carlos A. Coello Coello, *"Multi-Objective Particle Swarm Optimizers:A Survey of the State-of-the-Art,"* International Journal of Computational Intelligence Research, ISSN 0973-1873 Vol.2, No.3, pp. 287–308, 2006

[14] Carlo R. Raquel, University of the Philippines-Baguio, and Prospero C. Naval, Jr. University of the Philippines-Dilliman, *"An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization,"* GECCO'05, 2005

[15] Fatma Ben Taher, *"implementation of adaptation techniques on the H.264/AVC application,"* End of study memoire in computer science engineering, ENIS, 2010