République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Ecole Doctorale
Sciences et Technologies

Mémoire de *MASTERE*
NTSID

N°d'ordre: 2010 –130

MEMOIRE

Présenté à

L'École Nationale d'Ingénieurs de Sfax

En vue de l'obtention du diplôme de

MASTERE

Dans la discipline *NTSID*

Par

Mouna BEN SAID

A multi-constraints adaptation technique for embedded multimedia systems

Soutenu le 22 Juillet 2010, devant le jury composé de :

M.	**Mohamed ABID**	*Président*
Mme.	**Ilhem KALLEL**	*Membre*
M.	**Nader BEN AMOR**	*Encadreur*

République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Ecole Doctorale
Sciences et Technologies

Mémoire de *MASTERE*
NTSID
N°d'ordre: 2010 –130

A MULTI-CONSTRAINTS ADAPTATION TECHNIQUE FOR EMBEDDED MULTIMEDIA SYSTEMS

Mouna BEN SAID

Résumé : Le nombre croissant d'applications multimédia complexes a soulevé de nouveaux défis dans la conception des systèmes multimédia embarqués mobiles qui doivent pouvoir réagir à la limitation des ressources et les fluctuations de l'environnement externe pour fournir un service acceptable. Dans ce projet, notre objectif final est de permettre la continuité du service et garantir un niveau de qualité minimal. Nous avons développé une technique d'adaptation permettant à un système multitâche de s'adapter aux variations de la charge du processeur et la bande passante réseau en utilisant l'allocation des ressources partagées et la reconfiguration des applications. Les tests ont été réalisés sur l'application de compression vidéo H264/AVC. Nous générons des configurations de manière automatique à l'aide de l'Optimiseur MO-PSO. Les résultats expérimentaux ont montré l'efficacité de la méthode.

Mots clés : Système embarqué, application multimédia, adaptation, partage de ressource, MO-PSO, reconfiguration, H264/AVC

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 provide an acceptable service. In this project, our final goal is to enable the continuity of service and guarantee a minimal quality level. We developed a multi-constraints adaptation technique enabling a multi-task system to adapt to variations in both processor-load and network bandwidth resources using shared resource allocation and application reconfiguration. Tests were done on the H264/AVC video compression application. We generate the application configurations automatically using the MO-PSO optimizer. The experimental results have shown the effectiveness of the method.

Key-words: Embedded system, multimedia application, adaptation, resource sharing, MO-PSO, reconfiguration, H264/AVC

الخلاصة أدى العدد المتزايد من تطبيقات الوسائط المتعددة المعقدة إلى تحديات جديدة في تصميم النظام الجوال المضمن للوسائط المتعددة التي ينبغي أن تستجيب لقيود الموارد والتغيرات في البيئة الخارجية لتقديم خدمة مقبولة. في هذا المشروع ،هدفنا النهائي هو ضمان استمرارية الخدمة و الحد الأدنى للجودة . لقد قمنا بتطوير تقنية على التكيف التي تسمح للنظام تعدد المهام على التكيف مع التغيرات الطارئة على الضغط على المعالج وعرض النطاق الترددي لشبكة توزيع باستخدام الموارد المشتركة وإعادة تشكيل التطبيقات. وأجريت التجارب على تطبيق H264/AVC من ضغط الفيديو. قمنا بتوليد أنماط تلقائيا باستخدام محسن MO-PSO . وأظهرت النتائج التجريبية فعالية الأسلوب .

المفاتيح: النظام الجوال المضمن للوسائط المتعددة ,تطبيقات الوسائط المتعددة ,التكيف ,MO-PSO ,إعادة تشكيل, H264/AVC

Summary

Figure list

Table list

General introduction

The growing number of embedded multimedia systems has led designers and manufacturers to compete in areas like cost, hardware requirements, coding efficiency, and error resilience and recovery.

However, the rapid development of multimedia applications has raised new challenges in multimedia systems design due to their mobility: in order to provide a minimal acceptable service, it is of great importance to be able to react to resource limitations and external environment fluctuations. These include limitations on local resources imposed by weight and size constraints, concern for battery power consumption, unpredictable variation in network load, and lowered trust and robustness resulting from exposure and motion. For instance, multimedia applications such as streaming video, that use unreliable connections such as Internet, wireless networks, must scale their complexity according to the network bandwidth availability in order to deliver meaningful results, i.e. acceptable video quality of service (QoS).

The goals for maximizing QoS and minimizing complexity present an inherent conflict in mobile systems design methodologies: a high QoS needs a high utilization of system resources such as CPU cycles and leads consequently to high power consumption; a low QoS would utilize few resources and so consume low power, but yields 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 multi-constraints adaptation technique for embedded systems running CPU/bandwidth intensive multimedia applications. This technique considers adaptation 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 applications parameters according to shared resources allocations while maintaining a minimal service quality defined as a user preference.

As a case study, we consider a H264/AVC video encoder to apply these adaptations. We focus on the method of generating the configurations for complex applications exhibiting a large set of possibly tuned parameters. We also consider the frequency of reconfiguration as well as the coordination between different adaptation levels in order to maintain the system stability.

This report consists of three chapters. The first one is a state-of-the-art of adaptation in embedded systems. We present different adaptation approaches that have been proposed in the literature, and point out their benefits and limits. In the second chapter, we detail our proposed adaptation technique. We describe the bloc structure of our adaptive system and detail the different design steps. The third chapter presents a case study application for the implementation of our adaptation technique and the experimental results.

Chapter 1

Adaptation approaches: state of the art

1. Introduction

Embedded systems running soft real-time multimedia applications, such as image, audio, and video, are becoming an increasingly important computing platform. For example, we can already use a smart-phone to record and play video clips and use an iPAQ pocket PC to watch TV. Compared to conventional desktop and server systems, embedded systems are often subject to resource constraints such as limited battery life, narrow memory space, and environment variations such as sudden and unpredictable network load. Thus, they are asked to adapt internally to their limited computational and energy resources as well as to changes in the external environment in order to support multimedia quality of service.

Researchers have then focused on the integration of adaptation strategies in the design flow of embedded multimedia systems. There have been numerous research contributions in adaptation in both hardware and software (operating system, application) layers of mobile multimedia systems.

In this chapter, we present a state of the art of adaptation approaches in the different layers of embedded multimedia systems. We also present some adaptation works particularly related to video coding application which is the most treated application in the literature.

2. Adaptation approaches

2.1 Adaptation in the hardware layer

Due to their modest sizes and weights, small mobile devices have inadequate resources, lower processing capabilities, memory, storage and power as compared to desktop systems. These portable systems are mostly battery driven and oftentimes have to run for considerable time periods. The most serious limitation on these devices is the available battery life.

Therefore power management has become a critical issue for those systems. Tackling power dissipation and improving service times for these devices are crucial research challenges. [1]

Diverse efforts have been made to improve power management in mobile devices using hardware and software techniques. Dynamic Voltage scaling (DVS) [2] and dynamic power management (DPM) techniques have been widely studied as methods for optimizing the power consumption.

DVS has been a key technique in exploiting the hardware characteristics of processors to reduce energy dissipation by lowering the supply voltage and operating frequency. The DVS algorithms are shown to be able to make dramatic energy savings while providing the necessary peak computation power in general-purpose systems. However, for a large class of applications in embedded real-time systems like cellular phones and camcorders, this technique is not efficient. Indeed, the variable operating frequency interferes with their deadline guarantee mechanisms, which reduce DVS performance. To provide real-time guarantees, researchers [3] developed a class of novel algorithms called real-time DVS (RT-DVS) that consider deadlines and periodicity of real-time tasks, requiring integration with the real-time scheduler. The RT-DVS modify the OS's real-time scheduler and task management service to provide significant energy savings while maintaining real-time deadline guarantees.

The DPM is a design methodology that dynamically reconfigures an electronic system to provide the requested services and performance levels with a minimum number of active components or a minimum load on such components. The fundamental premise for the applicability of DPM is that systems (and their components) experience non uniform workloads during operation time. In this technique, a power-managed system is considered as a set of interacting power manageable components (PMC's) controlled by a power manager (PM). The fundamental characteristic of a PMC is the availability of multiple modes of operation that span the power-performance tradeoff.

With PMC's, it is possible to dynamically switch between high-performance high-power modes of operation and low-performance low-power ones.

The Power State Machine (PSM) model of the Strong ARM SA-1100 is shown in Figure 1. States are marked with power dissipation and performance values, edges are marked with transition times.

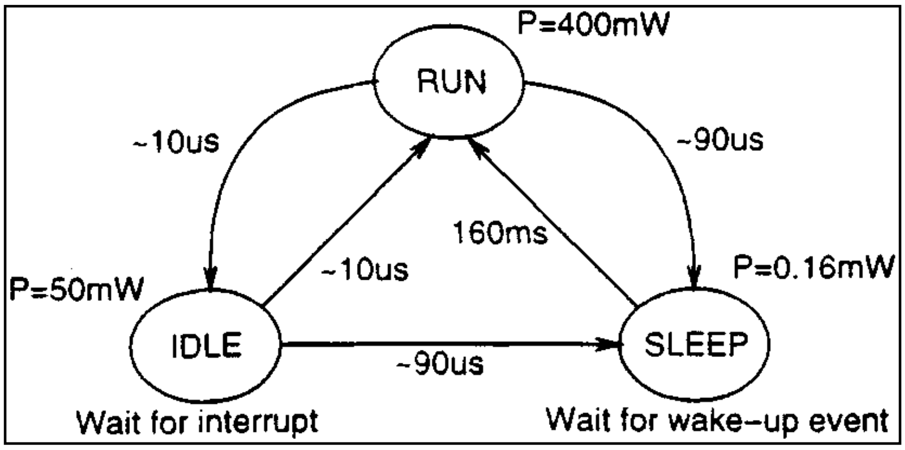


Figure 1. *The PSM model of the Strong ARM SA-1100*

DPM technique can be implemented with different schemes. Clock Gating: We consider first digital components that are clocked. This class of components is wide, and it includes most processors, controllers and memories. Power consumption in clocked digital components (in CMOS technology) is roughly proportional to the clock frequency and to the square of the supply voltage. Power can be saved by reducing the clock frequency (and in the limit by stopping the clock), or by reducing the supply voltage (and in the limit by powering off a component).

Other middleware frameworks such as Odyssey [4] present an application aware adaptation scheme which combines middleware notification and control with application adaptation to achieve good savings of the system total energy. The PARM framework [1], whereas, does not require the application to perform any energy adaptations. It presents a distributed middleware framework that is inherently power-aware and reconfigures itself to adapt to diminishing power levels of low-power devices.

2.2 Adaptation in the OS layer

The role of the OS in an embedded real-time adaptive system is to sense external events, monitor and allocate scarce resources. OS adaptation changes the policies of allocation and scheduling in response to application and resource variations. Many works have dealt with the operating system layer to provide predictable CPU allocation. In [5] [6] the managers of CPU resources provide performances guarantees in soft real time. Schedulers adapt the scheduling policy to handle the variations of application runtime [7] [8].

In [9] the authors present an energy-efficient soft real-time CPU scheduler for multimedia applications running on a mobile device. EScheduler seeks to minimize the total energy consumed by the device while meeting multimedia timing requirements. To achieve this goal, it integrates dynamic voltage scaling into the traditional soft real-time CPU scheduling: It decides at what CPU speed to execute applications in addition to when to execute what applications. EScheduler makes these scheduling decisions based on the probability's distribution of cycle demand of multimedia applications and obtains their demand distribution via online profiling.

2.3 Adaptation in the application layer

Application-aware adaptation is an essential capability of mobile systems. A technique that has been widely used is to change the application algorithms or parameter values to trade off output quality for resource usage. Indeed, researchers in the literature have focused on the improvement of video codecs' performance. Many of them are interested in optimizing the coding algorithms (reducing the computational time, improving quality). Many others focus on the adaptation of codec according to input data types. In our case, we are interested in the adaptation aspect. We give in this section an overview of some application-aware adaptation frameworks. Then we present a state of the art of adaptation techniques applied to video coding and particularly to the MPEG2 and H264/AVC standards.

2.3.1 Overview

Several projects recommend the adaptation in the application layer. For example, in [10] authors discuss the importance of a mechanism of task adaptation even in the presence of DVS or other power-reducing mechanisms, in order to make the best use of the available energy resources. They developed a framework called Energy-aware Quality of Service (EQoS) that can manage real-time tasks running on an embedded device and adapt their execution to maximize the system utility for a limited energy budget. They formulate energy adaptation of task sets into a tractable, optimal-selection problem. They consider a set of N random periodic tasks *T1…..TN* of which each task *Ti* has *mi* different QoS levels randomly generated as well. They formulate the basic energy adaptation problem as a selection of QoS levels combination under energy budget constraint, and use simple optimal algorithms and approximation heuristics for solving this problem.

In [11], Mesarina and Y. rather consider a specific multimedia application. They propose a method to reduce the energy for the “decoding MPEG” application using parameter modifications. [12] presents a QoS framework for interactive 3D applications where the QoS management relies on high-level QoS parameters (e.g. PSNR) of quality scalable 3D objects. This framework aims to guarantee the user specified interactive frame rate through degrading the 3D objects quality in such a way that minimal overall quality degradation over the scene is obtained.

Other middleware frameworks such as Odyssey [4] present an application aware adaptation scheme which combines middleware notification and control with application adaptation to achieve performance improvements.

2.3.2 Data-independent adaptation strategies

Many researchers have dealt with adaptation in the MPEG standard. In [13] S.C. Hsia proposes a rate control scheme which is based on an adaptive GOP that uses more P- and B-frame coding, instead of a static group. The main purpose of the algorithm is the dynamic modification of the GOP structure according to the temporal correlation between inter-frames. When a buffer under-flowing, a scene change, or a high accumulated error is detected, the algorithm switches from the prediction mode coding to the I-frame coding.

The patent [14] presents a scalable decoding technique that can dynamically adapt to resource constraints. It selects a scaling strategy that meets a complexity requirement coming from a Resource Manager. Each strategy is based on a set of scaling functions (e.g. scaling of adaptive B-frame, IDCT, motion compensation, embedded resizing).

In [15] authors present a method for quality aware frame selection for MPEG decoding under limited resources. They propose a frame skipping method to reduce the workload of the decoder when the processor fails to decode all frames timely. This method uses a frame priority assignment that assigns for each frame a priority to be decoded or skipped. They propose also a guarantee algorithm that takes as input the video stream, the available resources, and the priority values, and gives as output all frames which can be successfully decoded in time according to the system load.

2.3.3 Content-aware adaptation strategies

It is hard to conceive a universal algorithm that can perform well for all kinds of video contents. However, if important characteristics of a video sequence can be identified and utilized for adjusting various steps of video coding, one can design an adjustable algorithm that can tune its parameters to suit the video at hand.

Plenty of works in the literature have dealt with the adaptation in H264/AVC standard, in particular in the motion estimation (ME) block. Several motion estimation algorithms have been proposed where the complexity of the search is traded-off. They reduce the number of candidate blocks processed for each current block but without exploiting the statistics of the input data. On the contrary, a class of predictive algorithms exploits the spatial and temporal correlation that characterizes the input video. In [16] [17] and others, a technique that is commonly used consists of comparing the distortion in the image with a set of predefined thresholds to adjust some search parameters such as the number of reference frames (Nref) and the size of search window (SR). Indeed, Many researchers have focused particularly on the video motion characteristic [18] [19] [20] [21] [22]. They generally use a motion estimation algorithm that includes a pre-stage for an online analysis of motion activity in the input video. Based on this characteristic analysis, an online algorithm reconfigures dynamically the search parameters to allocate to the encoder the adequate computational resources. Some examples of such adaptation techniques are presented hereafter.

2.3.3.1 Window-follower approach

The high computational load and the relevant memory access requirements lead to high power consumption. Therefore the conventional FS approach is not suitable for battery-supplied devices where the low-power constraint is mandatory. [16] proposed a technique that exploits the input data variations to dynamically reconfigure the search-window size of an exhaustive block-matching search thus avoiding unnecessary computations and relevant memory accesses. It takes into account both spatial and temporal correlations of the motion vector (MV) field.

The motion activity of a scene directly relates to the values of the relevant MV field. In the case of high motion activity a large search-window size permits to get a MV that results in small estimation error. When the amount of motion in the scene is small searching over a large search area is not necessary. Based on these considerations, and taking into account the spatio–temporal correlation of real-world MV fields, it is possible to devise a window-follower motion estimator in which the search window size, for all the blocks of a frame, depends on the maximum displacement in the MV fields of the current and/or previous frame. This way, the size of the search window follows the MV field. The proposed algorithm consists of two steps.

Step 1: From the complete list of MV of the previous *(k _ 1)*th frame, compute the maximum horizontal and vertical displacement, hereafter called *D*, defined as

$$D = \max_{(t)} [d_t]$$

Being $d_t = \max [MVt_x, MVt_y]$

Step 2: Compute MV for all the blocks of the k^{th} frame by adopting an exhaustive search with a window displacement p_t; for the generic t^{th} block, sized according to the following mutually exclusive rules:

$$\boxed{\begin{array}{ll}
\text{(i)} & SAD_{min_{t-1}} \geq TH_1 \rightarrow F = 1 \text{ and } p_t = p_{max} \\
\text{(ii)} & TH_2 \leq SAD_{min_{t-1}} < TH_1 \text{ and } F = 1 \rightarrow p_t = 1 + \max(D, d_{t-1}) \\
 & TH_2 \leq SAD_{min_{t-1}} < TH_1 \text{ and } F = 0 \rightarrow p_t = 1 + D \\
\text{(iii)} & SAD_{min_{t-1}} < TH_2 \text{ and } F = 1 \rightarrow p_t = \max(D, d_{t-1}) \\
 & SAD_{min_{t-1}} < TH_2 \text{ and } F = 0 \rightarrow p_t = D
\end{array}}$$

Figure 2. *Window follower algorithm*

For the first frame *p = pmax*: SADmint_1 and dt_1 represent, respectively, the *SADmin* and the maximum MV displacement for the *(t _ 1)*th block of the current frame.

The thresholds used to evaluate the efficiency of the motion prediction are TH1 = 4096 and TH2= 2048 are. Practically, when a motion change occurs and the actual search size is not large enough, the increase of the prediction error and of the relevant *SADmin* is detected by TH1 (changement brusque de motion) that sets the search displacement to the maximum value. The threshold TH2, instead, is useful in case of gradual motion changes to avoid unnecessary search size increments. This work has reached the same FS performance while considerably reducing power consumption.

2.3.3.2 *A dynamic control of three ME search parameters*

[17] proposed a technique that optimizes 3 ME search parameters in the same control environment (3 algorithms combined together).

- Reference Frame Control algorithm

This control decides the maximum number of reference frames to be used for the ME of each 16x16 macroblock (MB) and its enabled sub-partitions. Figure 3 illustrates this first algorithm.

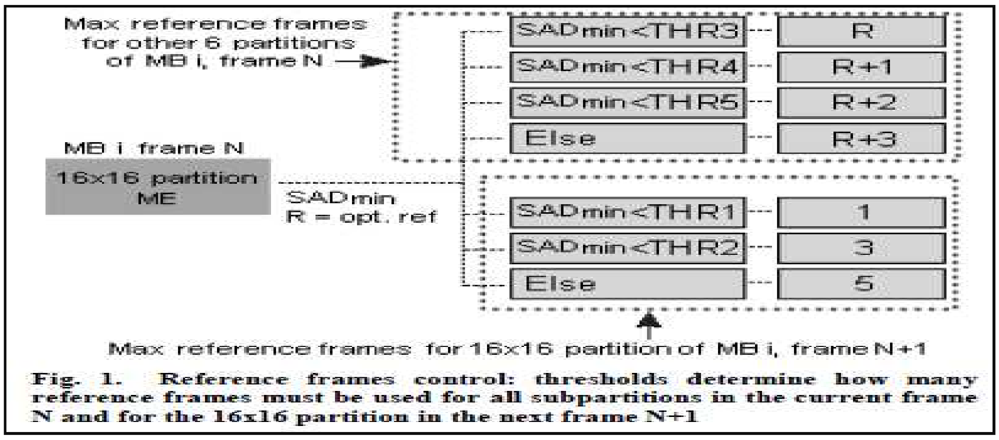


Fig. 1. Reference frames control: thresholds determine how many reference frames must be used for all subpartitions in the current frame N and for the 16x16 partition in the next frame N+1

Figure 3. *Reference Frame Control algorithm*

The thresholds determine how many reference frames must be used for all sub-partitions in the current frame N and for the 16x16 partition in the next frame N+1. The more *SADmin* increases the more the number of reference frames *R* increases too. The best number of reference frames depends on the dynamism of video input, i.e. static videos often need only one reference frame, and fast moving videos need more

- Block mode control algorithm

H.264/AVC can encode a MB using 7 possible block modes from 16x16, 16x8 …to 4x4. Figure 4 presents the proposed block mode selection algorithm where the more the SADmin increases the more additional smaller sub-partitions are enabled.

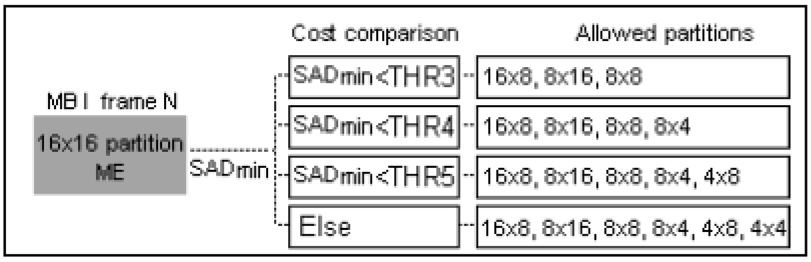


Figure 4. *Block mode selection algorithm*

- Search area control algorithm

There are algorithms that reconfigure the search area, for Full Search (FS) ME algorithm, according to the input signal statistics. The search displacement is configured on the basis of the MV field of neighboring blocks. Authors have made the porting of this technique to EPZS and UMHS ME algorithms.

2.3.3.3 A Region Of Interest based adaptation

K. Kanur and B. Li present in [22] a framework which allocates the computational power of the encoder adaptive to video contents, and also scales with the available battery power using a Region Of Interest (ROI) classification method. They combine the ROI-based encoding (video segmentation into ROI and background based on motion analysis) with a power scalable encoding parameter set selection process. This process allocates high computational resources to the slice group with ROI, i.e. coded with all partition modes and higher search range, and the static background regions (having little or no motion) can be coded with a restrictive parameter set by excluding smaller partition modes and using small search range.

2.4 Cross- layer Adaptation

The above adaptation techniques have been shown to be effective for both QoS provision and energy saving. However, most of them adapt only a single layer or two joint layers. For example, [23] presents an application-aware adaptation strategy of collaborative partnership between the operating system and individual applications to better cope with the problems induced by mobility in the context of shared data access.

More recently, some groups have proposed cross-layer adaptation frameworks in which different system layers adaptations are coordinated in order to fully exploit the adaptation benefits. Some of these cross-layer approaches [10] [24] adapt only at coarse time granularity, e.g., when an application joins or leaves the system. This infrequent adaptation is proven to be insufficient to deal with small changes in the system environment and processed data. Then, some other cross-layer adaptive frameworks have been proposed to adapt systems at both coarse and fine time granularities. We present hereafter two recent cross-layer adaptation approaches.

2.4.1 GRACE project

The GRACE project (Global Resource Adaptation through CoopEration) [6] [25] proposes a cross-layer adaptation framework that adapts multiple system layers at multiple time granularities. It addresses the conflict of adaptation scope and frequency through a hierarchical solution (c.f.Figure 5) which combines different adaptation levels:

- An infrequent expensive global adaptation that optimizes energy for all applications in the system. It only occurs occasionally at large system changes such as application entry or exit.
- A frequent inexpensive limited-scope per-application adaptation that optimizes for a single application at a time. It is invoked every frame, adapting all system layers to the application's current demands. An internal adaptation adapts only a single system layer and may be invoked several times per application frame.

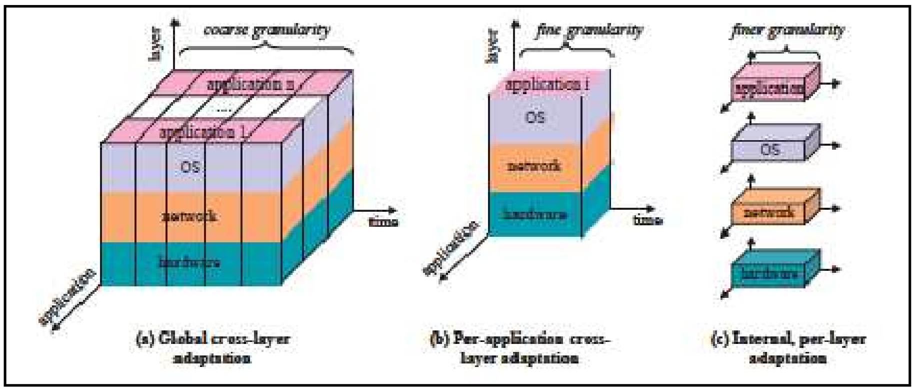


Figure 5. *GRACE adaptation hierarchy*

GRACE's first generation implementation, called GRACE-1 [6] coordinates the adaptation of the CPU speed in the hardware layer, CPU scheduling in the OS layer, and multimedia quality in the application layer, in response to system changes at both fine and coarse time granularities. It focused also on the coordination between these cross-layer adaptations in order to reap their full benefit. GRACE-1's focus was on cross-layer global adaptation, for which it showed significant energy benefits. It reported a few experiments with hierarchical adaptation in the CPU and scheduler, but showed only modest benefits over global adaptation when running multiple applications.

Later, GRACE demonstrates the benefits of the hierarchical adaptation through a second generation prototype, GRACE-2 [25]. This prototype implements a global adaptation in the CPU, application, and soft real-time scheduler; per-app adaptation in both CPU and application layers; and internal adaptation in the scheduler. It respects the constraints of CPU utilization and network bandwidth (assumed to be constant), while minimizing CPU and network transmission energy.

The fundamental difference between the GRACE-1 and GRACE-2 systems is that GRACE-2 is network-aware. It adds a network bandwidth constraint in the global and per-application controller and considers global and per-app application adaptations that are driven by the tradeoff in CPU time and network bandwidth usage. GRACE2 has demonstrated that the lack of any network awareness results in very modest benefits from the hierarchical adaptation while running multiple applications. It has also shown that in a network bandwidth constrained environment, per-app application adaptation yields significant energy benefits over and above global adaptation. However, we have to mention that these observations concerning network constraint were made by assuming a non-adaptive (simulated) network layer with fixed available bandwidth.

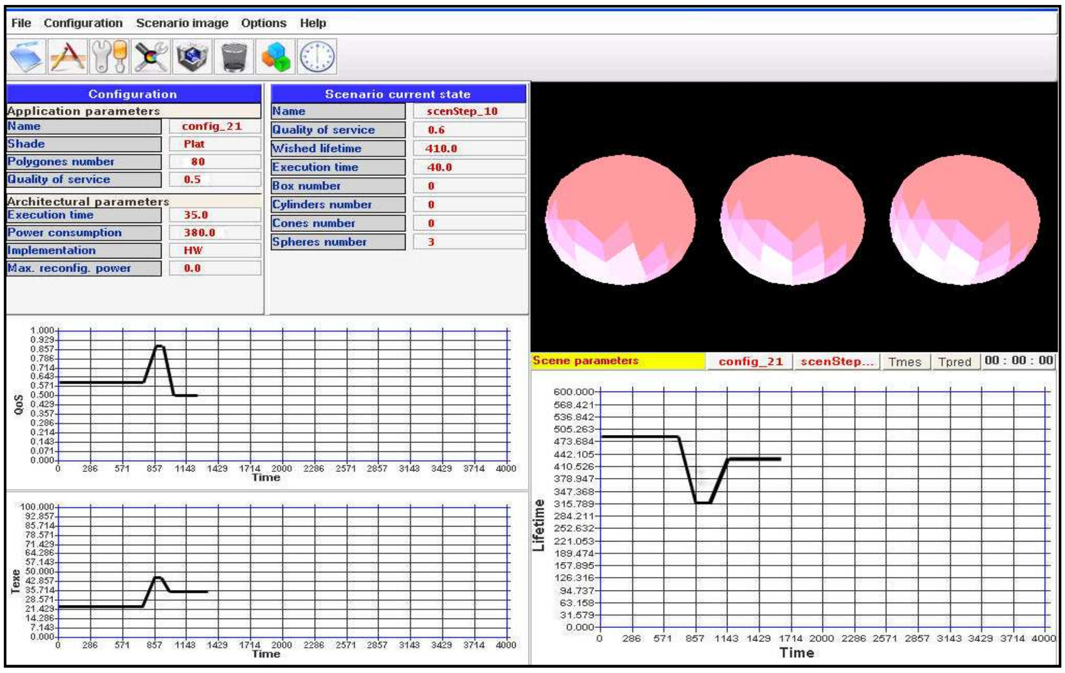
2.4.2 Adaptation approaches proposed in CES laboratory in cooperation with lab STICC

The previously described cross layer adaptation technique has limited HW adaptation since it manages CPU frequency and voltage which is not allowed for all processors. Besides, cross layer techniques were adopted in laptops or web servers which are not typically embedded systems. For those purpose, many issues were not discussed such as the cost of the adaptation technique that must match the limited resources of embedded systems. To address those problems, an adaptation technique was developed in the research unit CES (ENIS, Tunisia)[1] in cooperation with the lab STICC[2] (Lorient, France), particularly with the professor Jean Philippe Diguet. We present this technique in the following sub-sections.

[1] http://www.ceslab.org/

2.4.2.1 *Approach proposed by N. Ben Amor*

N. Ben Amor proposed in [26] a multi-constraints adaptation approach. It combines adaptation on HW level and application level. Adaptation on HW level is based on the selection, at run time, of a configuration from a set of previously characterized ones. Those configurations run different architecture schemes of the application. Each one has specific energy and performance levels. In general, high performance configuration is based on a processor enhanced with specific hardware circuits so that it will be less energy efficient than a configuration with a single processor. Adaptation based on application level modifies some application parameters to manage the desired QoS and the desired lifetime. This approach was validated by a high-level simulator using the 3D image synthesis as a case study. Figure 6 gives an overview of this simulator.



Configuration	
Application parameters	
Name	config_21
Shade	Plat
Polygones number	80
Quality of service	0.5
Architectural parameters	
Execution time	35.0
Power consumption	380.0
Implementation	HW
Max. reconfig. power	0.0

Scenario current state	
Name	scenStep_10
Quality of service	0.6
Wished lifetime	410.0
Execution time	40.0
Box number	0
Cylinders number	0
Cones number	0
Spheres number	3

Scene parameters | config_21 | scenStep... | Tmes | Tpred | 00 : 00 : 00

Figure 6. *A high-level simulator for adaptation in the 3D image synthesis*

The author has developed a QoS function that models the 3D image quality. He uses an accurate battery model (Rakhmatov model) that gives an accurate value of the system lifetime using consumed current values. The use of the battery lifetime as a constraint instead of the consumed energy is depicted with several works. Several configurations were built around the NIOS processor ranging from simple software configuration to high performance

[2] http://recherche.telecom-bretagne.eu/lab-sticc/

configuration. Power consumption models are used in the high level simulator to have accurate system output values (lifetime, QoS, and execution time).

2.4.2.2 Approach proposed by K. Loukil

The previously described approach was enhanced by K.Loukil in [27][28]. He presents a coordinated cross layer adaptation approach in the hardware, OS, and application layers for embedded multimedia reconfigurable systems (c.f. Figure 7). The proposed approach makes it possible to improve the quality of service while respecting the system constraints (real time, lifetime, QoS) and the user preferences (level of minimum QoS, lifetime). It also deals with the coordination between the different system layers. To address this problem, authors propose an additional middleware layer which comprises:

- a global manager which coordinates between the three layers in order to answer the great variations of the system constraints and user preferences. It chooses the adequate configuration of the system starting from a configuration base.
- a local manager that intervenes only in the application and OS layers. It is set up to essentially control the respect of the system real time constraint. It can also question the global manager if it does not manage to find a solution to solve a problem of timing constraints violation.

In the hardware layer, adaptation is done using one of the different application implementations called configuration that are previously set up and characterized. Those configurations vary from a pure SW configuration (called config_sw) to a mixed implementation with several HW components (called config_hw). As config_sw involves only CPU resources, it will consume less energy than config_hw but will less performing. So those, the system can choose the adequate configuration according to the constraints and the user preferences.

Adaptation in application layer is performed by modifying the application parameters or algorithms according to the imposed constraints. A 3D object synthesis application is considered when implementing this work. The parameters to tune are the number of triangles composing the 3D object and the shading algorithm (Flat, Gouaud). This technique is implemented on the µC/OSII and run on a Stratix FPGA.

The third adaptation level is made in the operating system layer in order to allocate necessary CPU time for each task. In [29], we propose a technique for the control of application timing constraints that is hooked into the OS kernel. It deals with the problem of task deadline exceeding to adapt to data variations which may cause real time constraints violation. This technique monitors tasks execution by means of watchdogs in order to detect possible deadline violation. The detection result is then used to tune the target application parameters in order to satisfy the real time constraints for the further computation iterations.

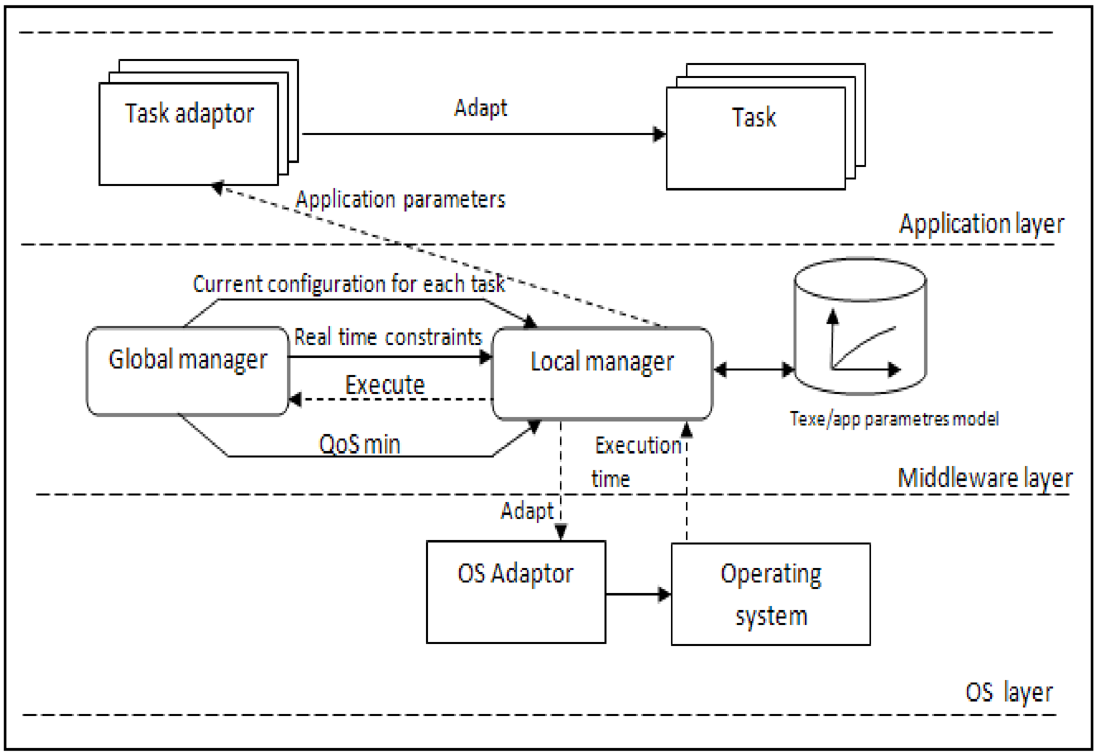


Figure 7. *Structure of a cross-layer adaptation approach*

3. Conclusion

With reference to the state of the art presented in this chapter, different adaptation approaches were proposed in the literature. They consider different constraints (mono/multi constraints), different system layers combinations, and different objectives (minimize total system energy, maximize QoS…). However, most of these approaches were not dedicated to embedded systems. They have been developed for example for PCs or web servers. Adaptation techniques that were developed for embedded systems were, to our knowledge, limited to a single constraint, mainly the remaining energy of the system. Whereas, as embedded systems are usually mobile, they must adapt to an increasing number of constraints related to the external environment, the nature of the target applications, and the user choices.

To address this problem, a new cooperation research project called “ConcEption de Systèmes Adaptatifs multi-niveaux pour le traitement Multimédia Embarqué” (Césame) has started. The purpose of this project is the design of an adaptive architecture for a complex multimedia application (video compression and 3D image synthesis). Adaptation should satisfy many constraints such as energy, CPU time, network bandwidth, QoS, and user preferences.

Our present work is included in this project. We propose a multi-constraints adaptation technique that will be applied to the video compression application. In the next chapter, we give details of the proposed technique.

Chapter 2

Design of a multi-constraints adaptation technique

1. Introduction

As we mentioned in the previous chapter, the present work is involved in the "Césame" project which deals with adaptation in embedded multimedia systems. In this chapter, we start by introducing the "Césame" project in order to clarify the position of the present work.

Then, through observations of the first chapter state-of-art, we could identify a number of important issues in the area of adaptation as well as valuable techniques enabling high adaptation performance. In a second section, we summarize these observations and explain our contribution through an adaptation technique.

In the next sections of this chapter, we describe the proposed technique and detail the design strategy that we followed.

2. The CESAME project

"Césame" is a cooperation project (CMCU) between the research unit CES (ENIS, Tunisia) and the lab STICC (Lorient, France). The purpose of this project is the set up of a design methodology of adaptive and robust systems supporting a variety of complex multimedia and concurrent applications. It will be validated by a demonstrator based on Xilinx FPGA technology. The target applications are the H.264 video compression combined to the OPEN GL ES 3D image synthesis application. These complex applications involve many problems that must be resolved. On the one hand, the space of the algorithmic configurations is important because each of the applications has its own process and its large number of tunable parameters. This big solutions space imposes the use of complex adaptation with important cost. Therefore, the study of the compromise robustness/cost of the adaptation must be considered in a detailed way.

The choice of the application {3D+H.264} emanates from the improvements in the field of media applications which are being based on different media formats coming from different sources. Such composite applications are being used in different domains like military, medicine and Remote assistance.

The design of a system that executes such complex applications puts further design constraints like the choice of hardware accelerators, the memory management, and the energy consumption. For this purpose, high performance architecture capable of executing those applications efficiently must be set up. It is based on a multiprocessor architecture using specific computation units (coprocessor, hardware accelerators). An RTOS[3] is used to manage the complexity of both the application and the architecture. A camera is used for image capture and a network connection is used for video and 3D animations transmission.

This architecture is flexible and can run a set of hardware configurations using the technique of the dynamic reconfiguration allowed by the Xilinx FPGA.

3. Contribution

Our master project deals with the set up of an adaptation technique applied to the H.264 application. This technique is applied in the application level with no hardware support. The adaptation is based on the modification of algorithmic parameters of a purely software H.264 coder.

The essence of our proposed adaptation technique is that it considers adapting an embedded system to both internal (e.g. CPU load) and external (e.g. available Bandwidth) resources. It is derived from observations of the state-of-the-art adaptation models.

We consider adapting a multi-task system to resources change by scaling its QoS level according to resources availability. In fact, QoS level reduction has an influence on CPU utilization. Indeed, by selectively reducing the QoS level provided to individual tasks of a multi-task system, we can achieve a reduced total system load [10].

[3] Real Time Operating System

Scaling the system overall QoS level can be achieved by numerous techniques as it was presented in the state of the art. For instance, it can be obtained by changing the application computational complexity through algorithms and parameters modification in the application layer, and adapting scheduling policies to resources variation in the OS layer.

Among all these techniques, multiple previous works have shown the effectiveness of application adaptation and the importance of integrating network adaptation in multi-application network constrained systems. Authors in [4] show that application-aware adaptation offers the most general and effective approach to mobile information access. The GRACE project has shown in [17] the importance of both per-app application and network adaptation. It has also demonstrated that while running multiple applications the lack of any network awareness results in very modest benefits from the hierarchical adaptation (global and local adaptation). It has also shown that in a network bandwidth constrained environment, per-app application adaptation yields significant energy benefits over and above global adaptation.

Therefore, since our target system is network bandwidth constrained and has to run multiple concurrent applications, 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 considers adapting an embedded system not only to internal resources such as CPU time, but also to external environment such as network load fluctuation. It manages adaptations to different resources at application level at different time granularities. The advantage of application level adaptation is that it is portable to any platform unlike other techniques such as DVS which is not supported by all processors.

Yet, it is obvious that when designing such application-aware adaptive technique, target applications characteristics have to be taken into account. As we mentioned before, our target system is an embedded system running multimedia applications such as video compression and 3D synthesis. Such systems are sensitive to dynamically varying and multiple resource constraints (e.g., variations in available CPU time, and bandwidth) which affect the perceived quality and consequently the user's satisfaction. Hence, they must provide a minimum QoS in order to guarantee the continuity of service in the presence of fluctuating environment. Fortunately, the soft real-time nature of multimedia applications offers more opportunities than others for QoS trade-off. For example, video codecs, such as H264/AVC, are highly configurable. They offer a large number of possibly tuned parameters to encode and decode videos with different compression ratio, different computational complexity, and variable output quality.

According to the above observations on multimedia applications, we had the idea to profit from their high configurability to achieve different QoS levels and so better management of QoS/resources availability trade-off. 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.

Nevertheless, when adapting multimedia systems using application reconfiguration we face a key problem which is their operational control, i.e. QoS-levels definition and selection. 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 [26] and [27] define hardware and software configurations for a 3D synthesis application. These configurations yield a very limited number 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. Besides automation, this method enables the generation of non-dominated QoS-levels and thus obtaining a maximized system output quality.

4. Work flow

Our adaptation is based on a tradeoff between quality of service (reflects the user satisfaction), available network bandwidth (limit for the output bit rate), and real time constraint (limit for the CPU time allocated for each application). This implies:

- the existence of various application configurations offering different levels of QoS and complexity
- the ability of the system to share the available resources among competing tasks.
- the ability of the system to move from one configuration to another dynamically in response to changes in system load and bandwidth availability.

Our work flow is composed of two main steps:

- Offline characterization step (figure 8): we identify the different operational modes of the applications (configurations). This step is based on a study of the target applications in order to select the set of attributes that are more likely to influence the compromise QoS/ bandwidth/CPU time (configuration parameters). Those attributes are purely software parameters selected from the list of the application configuration parameters. We also need to select the set of parameters that will be considered to represent the QoS of the application (QoS parameters).

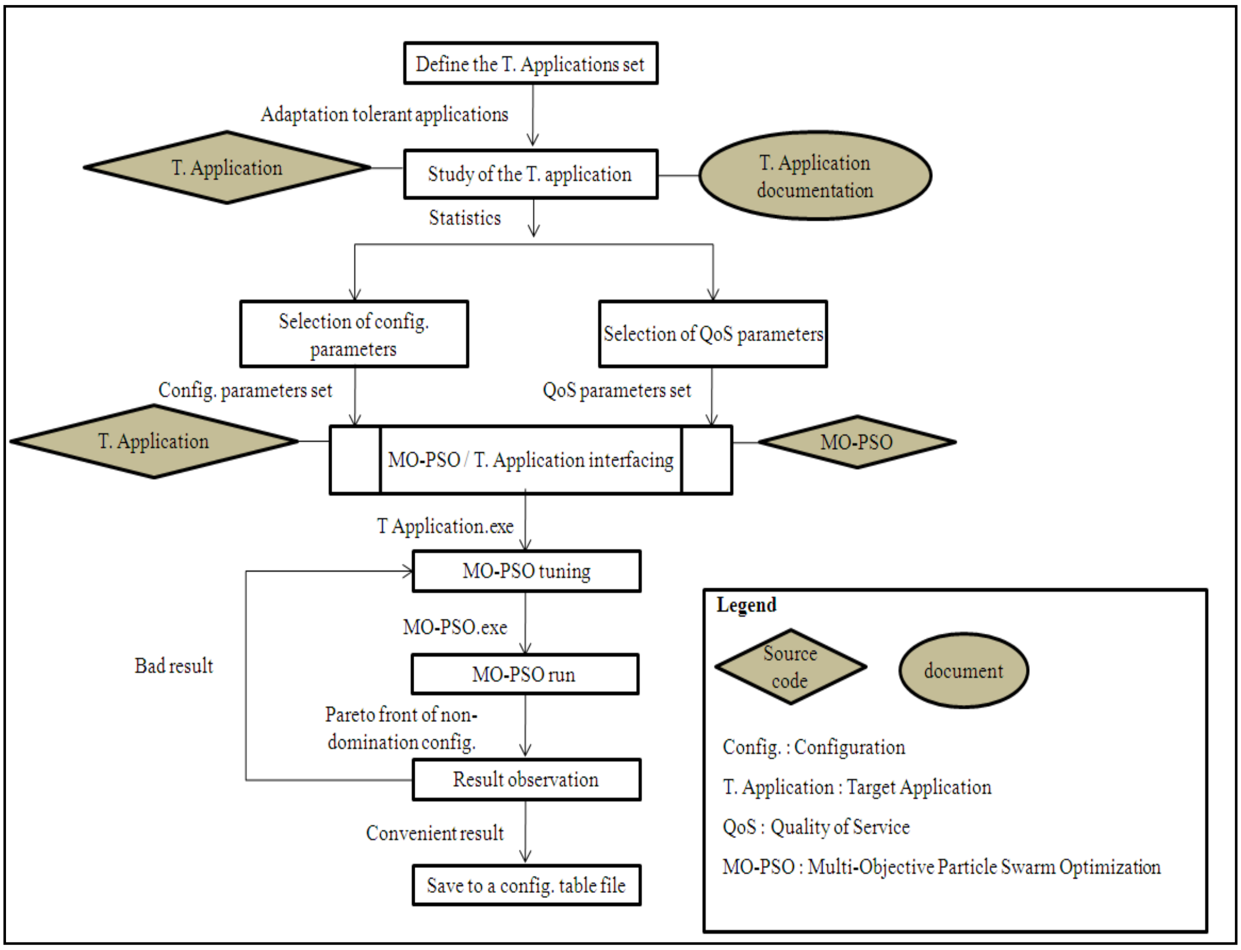

Define the T. Applications set
Adaptation tolerant applications
T. Application
Study of the T. application
T. Application documentation
Statistics
Selection of config. parameters
Selection of QoS parameters
Config. parameters set
QoS parameters set
T. Application
MO-PSO / T. Application interfacing
MO-PSO
T Application.exe
MO-PSO tuning
MO-PSO.exe
Bad result
MO-PSO run
Pareto front of non-domination config.
Result observation
Convenient result
Save to a config. table file
Legend
Source code
document
Config. : Configuration
T. Application : Target Application
QoS : Quality of Service
MO-PSO : Multi-Objective Particle Swarm Optimization

Figure 8. *Work flow 1: offline characterization step*

The result of parameters selection step is the input of the composite step of automatic configurations generation using the MO-PSO optimizer. This latter step consists of interfacing the MO-PSO with the target application and tuning its parameters to generate at the end executable files for both the optimizer and the application. Finally, we run the MO-PSO to generate the non dominated configurations. We observe the result to decide whether to keep it as a configuration table or to reject it, re-tune the optimizer and generate another pareto set.

- Online adaptation step (figure 9): we conceive an online adaptation platform whose objective is to maximize the system QoS under CPU time and network bandwidth constraints. Adaptive applications have to get reconfigured in order to meet these constraints.

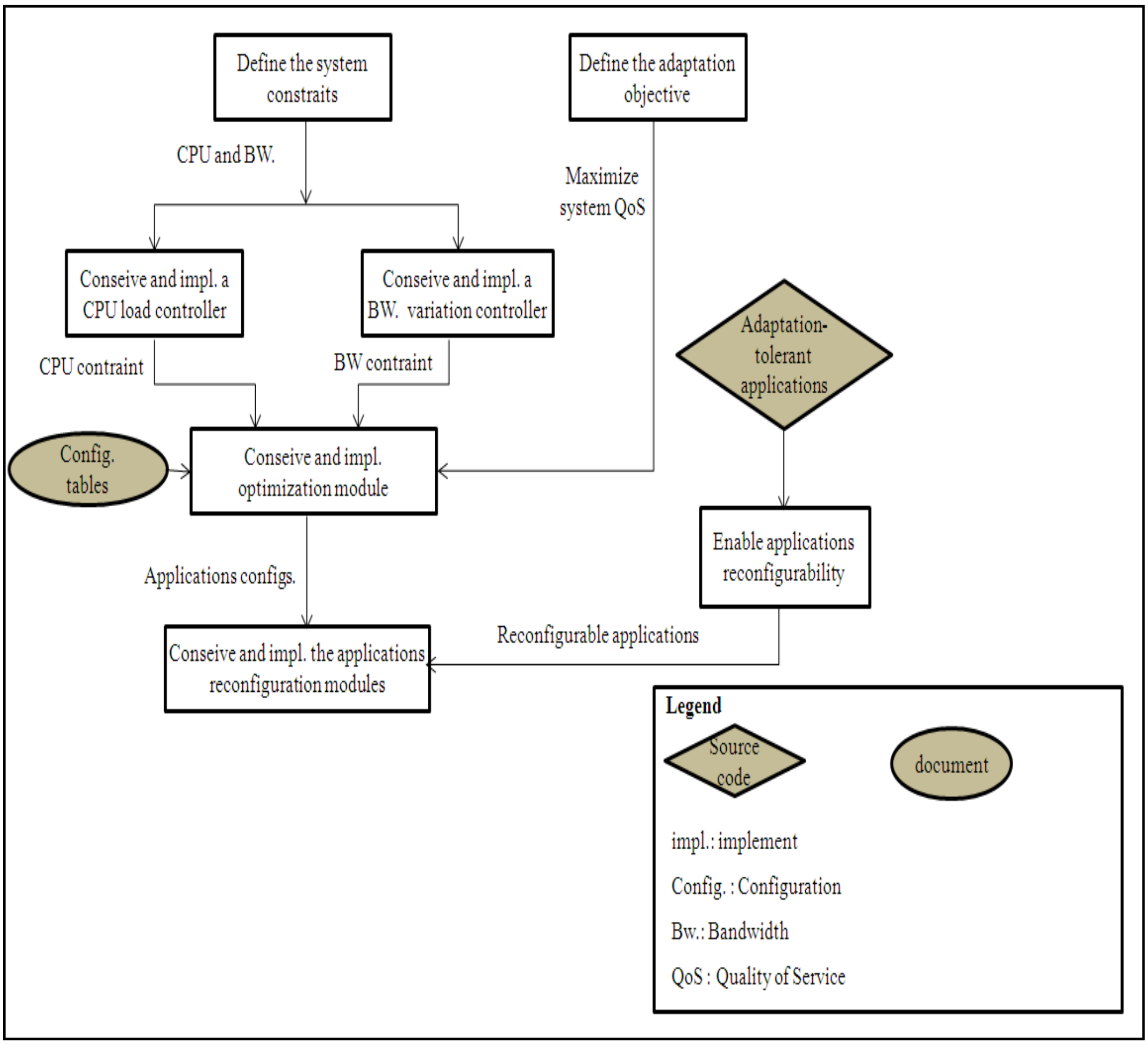

Define the system constraints
Define the adaptation objective
CPU and BW.
Maximize system QoS
Conseive and impl. a CPU load controller
Conseive and impl. a BW. variation controller
Adaptation-tolerant applications
CPU contraint
BW contraint
Config. tables
Conseive and impl. optimization module
Enable applications reconfigurability
Applications configs.
Reconfigurable applications
Conseive and impl. the applications reconfiguration modules
Legend
Source code
document
impl.: implement
Config. : Configuration
Bw.: Bandwidth
QoS : Quality of Service

Figure 9. *Work flow 2: online adaptation step*

For more understanding, let’s have an example of real scenario. Consider a mobile device running a video application. When it enters an area of low network coverage, its high bandwidth connectivity is lost. Being adaptive, the video application is notified by this change. It responds by reconfiguring itself to deliver a worse video with lower bit rate. When a considerable improvement in the bandwidth connectivity is detected, the application is also notified to revert to its best previous configuration.

Therefore, different modules have to be defined in our work flow. First, we conceive and implement resource controllers which monitor CPU and bandwidth variations. Second, we conceive and implement an optimization module that selects from configuration tables a combination of application configurations that best meets the system objective under the resource constraints coming from the resource controllers. This module has to consider fairness among competing tasks when allocating shared resources. The last step is to reconfigure the applications using the selected configurations. For this purpose, we first need to develop adaptive applications that are dynamically reconfigurable. Then, we conceive and implement application-specific reconfiguration modules which reconfigure each adaptive application according to the selected configuration.

5. The multi-constraints adaptation technique

In this section we describe the structure of our multi-constraints adaptation technique. We present the different design steps that we followed.

5.1 Global overview

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 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 10.

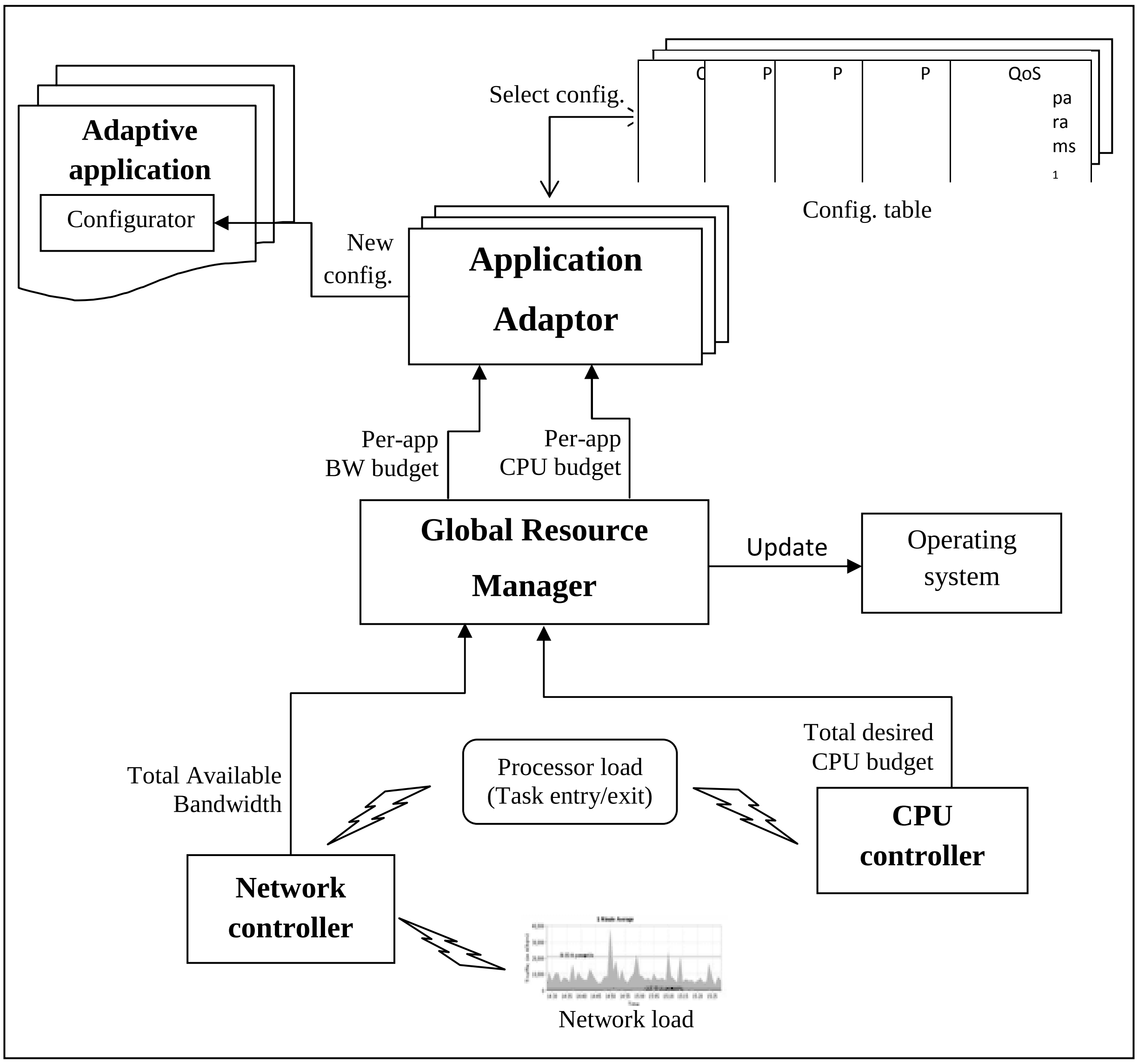


Figure 10. *Structure of the multi-constraints adaptation technique*

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 selector picks a configuration from a pre-established configuration table that best responds to resources allocated budgets. After that, an application-specific Configurator dynamically reconfigures the adaptation choices within the application. Thus, each application needs to have a corresponding Configurator.

In such structure, resource management modules would treat applications generically. Per-application adaptors would be entirely responsible for differential handling of applications.

5.2 Problem formulation

Formally, we describe our adaptive system as follows. Let's have a set of n periodic tasks $T_1\ldots\ldots T_n$ running concurrently on the system. Each task T_i has m_i different configurations, that we also call QoS levels and are defined as $L_{i,1}\ldots\ldots L_{i,mi}$. For each configuration $L_{i,j}$ we specify the task real-time characteristics. Let $P_{i,j}$ be its period, and $C_{i,j}$ 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 $B_{i,j}$ and QoS value $Q_{i,j}$. Let *Bw* be the available network bandwidth and *Pbudi* the CPU budget required by each task *i*.

If we want to adapt a mono-task system, we simply need to select the configuration of the running application that best satisfies the CPU time constraint (i.e. $C_{i,j} \leq P_{i,j}$), and the bandwidth availability (i.e. $B_{i,j} \leq Bw$), while maximizing the individual application QoS.

However, since we consider a multi-task environment, then the problem becomes very complex. In this case, the purpose of our adaptation model will be the selection of a combination of per-task QoS-levels L1,j…… Ln,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:

$$xij \begin{cases} 1 & \text{the } j^{th} \text{ configuration of the } i^{th} \text{ application is selected} \\ 0 & \text{else} \end{cases}$$

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

$$\begin{cases} \text{Maximize } \left(\sum_{i=1}^{n}\sum_{j=1}^{mi} Qij * xij\right) & (\text{E } 1) \\ \text{Subject to:} & \\ \sum_{i=1}^{n}\sum_{j=1}^{mi} \frac{Cij}{Pij} * xij \leq 1 & (\text{E } 2) \\ \sum_{i=1}^{n}\sum_{j=1}^{mi} Bij * xij \leq BW & (\text{E } 3) \\ \sum_{j=1}^{mi} xij = 1 \quad \forall\ i = 1,2,\ldots,n & (\text{E } 4) \end{cases}$$

We propose a simple method to solve this complex problem. It is to 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 (E2,E3). Then, each application has a specific adaptor which is responsible for selecting the best configuration that maximizes the quality and meets the allocated budgets. This way, we guarantee that resource constraints are met and the overall system quality is maximized, since we have maximized the local quality of each application.

At this level, two questions arise: how do we share available resources among competing tasks? And how do we define the application configuration set that facilitates the step of the best configuration selection? To answer the first question, we define a resource sharing algorithm that enables fair resource budgets allocation for adaptation purpose. As for the second question, we use a multi-objective optimization algorithm that:

$$\begin{cases} \text{Max (Qi,j)} \\ \text{Min (} C_{i,j} \text{)} \\ \text{Min (} B_{i,j} \text{)} \end{cases}$$

This algorithm should give us the optimal, non-dominant solutions, i.e. the set of application configurations that generate the best QoS for a given processing time and bit rate. Then, what remains is just seeking for a configuration, among the generated ones that best meets both of real time and bandwidth constraints.

In the next sections, we detail both of the two algorithms we've just mentioned.

5.3 The Global Resource Manager

The ability to execute multiple applications concurrently on a mobile system is increasingly becoming essential. Although this ability is taken for granted on desktops, there continues to be disbelief about its value in mobile systems. This doubt is fueled by the popularity and dominance of mobile devices executing only one application at a time. However, the sharp competition between actors in the field of mobile systems is inducing new services including the multi-tasking.

The multi-tasking presents a new challenge for embedded systems. In fact, to effectively accommodate concurrent applications, we need to control their use of limited resources. For this purpose, we have established in our system some resources monitors and a global resource manager. While designing the GRM, we face the problem of fairly dividing a scarce resource among a set of competing tasks, each of whom has an equal right to the resource, but some of whom demand fewer resources than others. How, then, should we divide resources amongst tasks to preserve the system global property called "fairness"? And how do we include adaptation mechanism in the resource sharing technique?

5.3.1 CPU adaptation

The basic idea of the CPU adaptation is to adjust the CPU allocation for each task based on a schedulability test using task execution time and period.

Let's consider a set of tasks *T1…Tn* and their time constraints. Let $t_1 \ldots t_n$ be their processing times and $p_1 \ldots p_n$ be their periods.

The schedulability analysis is the study of tasks' feasibility using the scheduling policy used. In other words, it is the fact of checking that all task jobs meet their timing constraints for any trajectory of the system task model. Then, a schedule is said feasible if all tasks resources and timing constraints are met.

We assume a set of tasks that are scheduled by the earliest deadline first (EDF) algorithm. It is a preemptive scheduler based on a dynamic task priorities assignment (the closer the task deadline, the higher the task priority). Under these assumptions, Liu and Layland give in [30] a necessary and sufficient condition for the tasks to be schedulable based on processor utilization. The task set is schedulable if and only if:

$$\sum_{i=1}^{n} \frac{ti}{pi} \leq 1 \qquad \text{(E5)}$$

5.3.2 Bandwidth adaptation

Our bandwidth adaptation is dedicated essentially to deal with the problem of video quality control for streaming video over drastically changing networks. In fact, the frame rate of the video streaming is affected by the network bandwidth availability.

The final aim of this network adaptation mechanism is to guarantee the continuity of service with acceptable quality despite variations in network bandwidth. Thus, 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.

5.3.3 Resource sharing technique

To establish the resource sharing technique used by the GRM bloc, we should first define the notion of fairness. Then, we give a brief overview of some resource sharing algorithms proposed in the literature, to arrive after that to our proposed algorithm designed for CPU and bandwidth allocation.

5.3.3.1 Notion of fairness

Researches in the literature have dealt with fairness as a system-wide property. Authors in [31] represent the notion of fairness as follows. Consider a set of *N* flows, competing for a single shared resource R. Each flow *i* is assigned a weight *wi* indicating its relative right to share the resource. For example, weights in a best-effort network are typically the same for all flows. Let *di* be the demand of flow *i* for the shared resource, and *ai* be its allocated resource. The normalized demand *Di* of flow *i*, which is the fractional share of the demanded resource, is defined as follows:

$$Di = di/R \qquad \text{(E 6)}$$

The normalized allocation *Ai* of flow *i*, which indicates the fractional share of the allocated resource, is defined as follows:

$$Ai = ai/R \qquad \text{(E7)}$$

Thus, given the resource consumption rates demanded by the flow, the notion of fairness specifies how to distribute the allocated fractional share of the resource to each flow. In other words, it defines a particular rate of resource consumption for each flow. Authors in [31] represent any given notion of fairness as a function as follows:

$$[Ai] = F(C, [Di], [wi]) \qquad \text{(E8)}$$

Where C is the constraint imposed on the system, [Ai], [Di], and [wi] are the vectors of, respectively, the normalized allocations, normalized demands, and flows weights. The function F is different for different notions of fairness.

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 proportional fairness [32].

5.3.3.2 Max-min fair share

A sharing technique that is widely used in practice is max-min fair share, also called “progressive filling”. Intuitively, a fair share allocates a user with a "small" demand what it wants, and evenly distributes unused resources to the "big" users.

Formally, we define max-min fair share allocation to be as follows:

- Resources are allocated in order of increasing demand
- No source gets a resource share larger than its demand
- Sources with unsatisfied demands get an equal share of the resource

The “max-min” calling comes from the fact that this allocation maximizes the share of sources with minimum demand, thus, gives relative priority to smaller sources.

This formal definition corresponds to the following operational definition. Consider a set of sources 1, ..., n that have resource demands x1, x2, ..., xn. Without loss of generality, order the source demands so that x1 <= x2 <= ... <= xn. Let the server have capacity C. Then, we initially give C/n of the resource to the source with the smallest demand, x1. This may be more than what source 1 wants, perhaps, so we can continue the process. The process ends when each source gets no more than what it asks for, and, if its demand was not satisfied, no less than what any other source with a higher index got.

For better understanding of the algorithm, let’s have the following example. Consider a set of four sources with demands 2, 2.6, 4, 5, and a network resource with a capacity of 10Mbps. We compute the fair share in several rounds of computation. In the first round, we tentatively divide the resource into four portions of size 2.5. Since this is larger than source 1's demand, this leaves 0.5 left over for the remaining three sources, which we divide evenly among the rest, giving them 2.66... each. This is larger than what source 2 wants, so we have an excess of 0.066..., which we divide evenly among the remaining two sources, giving them 2.5 + 0.66... + 0.033... = 2.7 each. Thus, the fair allocation is: source 1 gets 2, source 2 gets 2.6, sources 3 and 4 get 2.7 each.

5.3.3.3 Max-min weighted fair share

The previous definition of fairness is based on maintaining high budget for the sources with smallest demands. However, we may need in some cases to favor some sources over the others. That’s why the max-min fair share concept has been extended to include weights by defining the max-min weighted fair share algorithm.

The essence of this weighted algorithm is that it gives some sources a bigger share than others. We need to associate weights w1, w2, … wn with sources 1, 2, ..., n, which reflect their relative resource share. This algorithm is defined as follows:

- Resources are allocated in order of increasing demand, normalized by the weight
- No source gets a resource share larger than its demand
- Sources with unsatisfied demands get resource shares in proportion to their weights

The following example shows how to achieve this in practice. Let’s compute the max-min fair allocation for a set of four sources with demands 4, 2, 10, 4 and weights 2.5, 4, 0.5, 1 when the resource has a capacity of 16.

The first step is to normalize the weights so that the smallest weight is 1. This gives us the set of weights as 5, 8, 1, 2. We can now pretend that the number of sources, instead of being 4, is 5 + 8 + 1 + 2 = 16. We therefore divide the resource into 16 shares. In each round of resource allocation, we give a source a share proportional to its weight. Thus, in the first round, we compute C/n as 16/16 = 1. In this round, the sources receive 5, 8, 1, 2 units of the resource, respectively. Source 1 gets 5, and only needs 4, so we have 1 unit extra. Similarly, source 2 has 6 units extra. Sources 3 and 4 are backlogged, since their share is less than their demand. We now have 7 units of resources which have to be distributed to sources 3 and 4. Their weights are 1 and 2, and the smallest weight is 1, so there is no need to normalize weights. We give source 3 an additional 7 × 2/3 units (since its weight is 1), and source 4 an additional 7 × 2/3 units (since its weight is 2). This increases source 4's share to 2 + 7 × 2/3 = 6.666 units, which is more than it needs. So we give the excess 2.666 units to source 3, which finally gets 1 + 7/3 + 2.666 = 6 units. The final shares are, therefore, 4, 2, 6, 4. This is the max-min weighted fair share allocation.

5.3.3.4 Proposed adaptation-aware resource sharing algorithm

The sharing algorithms we have previously described 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 this purpose, we were brought to define a custom resource sharing algorithm that supports our adaptation mechanism. On the one hand, this algorithm 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. On the other hand, it guarantees that adaptive tasks are allocated the minimum necessary resource budget enabling them to achieve their minimal acceptable quality. The remaining amount of resource is divided among all adaptive tasks proportionally to their weights.

- **The algorithm**

We define for each task two types of resource budgets: a minimum resource budget, *minR_Bgt*, and an adaptive resource budget, *adaptiveR_Bgt*, if the task is adaptive. The *adaptiveR_Bgt* is the difference between the task total allocated budget, *R_Bgt*, and its *minR_Bgt*. We also define a total minimum budget, *totMinR_Bgt*, and a total adaptive budget, *totAdaptiveR_Bgt*. The former is the sum of *minR_Bgt* of all running tasks. It is used to check the worst case feasibility of a task set at every new task entry event. The latter is the sum of *adaptiveR_Bgt* of all adaptive tasks. Adaptive task weight is defined as a function of task *adaptiveR_Bgt* proportion of the *totAdaptiveR_Bgt*. The total budget, *totR_Bgt*, is the sum of all running tasks *R_Bgt*s.

The GRM performs the resource sharing at every task entry, task exit, or bandwidth variation event. The two first events trigger both CPU and bandwidth budgets re-allocation, while the latter requires only bandwidth budgets adjustment. We give hereafter the algorithms triggered by each of the three events. We note that these algorithms are performed with coordination between the resources controllers and the GRM. For generality purpose, we use the letter “R” to designate the resource. The algorithms fit both CPU and bandwidth resources.

1) *Task Entry (R, maxR, minR)*

***Algorithm*:**

```
Begin
    $MinR\_Bgt \leftarrow \frac{minR}{R}$
    $TotMinR\_Bgt \leftarrow totMinR\_Bgt + minR$

    If ($totMinR\_Bgt > 1$) Then
        Reject new task
        Exit
    EndIf

    $R\_bgt \leftarrow \frac{maxR}{R}$
    $totR\_Bgt \leftarrow totR\_Bgt + R\_bgt$

    If (task is adaptive) Then
        $AdaptiveR\_Bgt \leftarrow R\_bgt - minR\_Bgt$
        $totAdaptiveR\_Bgt \leftarrow totAdaptiveR\_Bgt + adaptiveR\_Bgt$
    Else
        $AdaptiveR\_Bgt \leftarrow 0$
    EndIf

    For each adaptive task
        $R\_bgt \leftarrow R\_bgt - \frac{adaptiveR\_Bgt}{totAdaptiveR\_Bgt} * (totR\_Bgt - 1)$
        $AdaptiveR\_Bgt \leftarrow R\_bgt - minR\_Bgt$
        $maxR \leftarrow R\_bgt * R$
    EndFor

    $totR\_Bgt \leftarrow 1$
    $totAdaptiveR\_Bgt \leftarrow 1 - totMinR\_Bgt$
End
```

***Description*:**

Our algorithm is performed in a single iteration through the running task set.

When a 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.

The new task undergoes an admission test using *totMinR_Bgt*. 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, it is accepted. If the new task entry is performed successfully then 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 an *R_bgt* proportional to its defined weight. At the end, the total allocated resource budget and the total adaptive resource budget are updated.

2) *Task Exit (idx)*

Algorithm:

```
Begin
    totR_Bgt ← totR_Bgt - R_bgt
    totMinR_Bgt ← totMinR_Bgt - minR_Bgt
    If (task is adaptive) Then
        totAdaptiveR_Bgt ← totAdaptiveR_Bgt – adaptiveR_Bgt
    EndIf
    Task deleted
    For each adaptive task
        R_bgt ← R_bgt - $\frac{adaptiveBgt}{totAdaptiveBgt}$ * (totR_Bgt - 1)
        AdaptiveR_Bgt ← R_bgt – minR_Bgt
        maxR ← R_bgt * R
    EndFor
    totR_Bgt ← 1
    totAdaptiveR_Bgt ← 1 – totMinR_Bgt
End
```

***Description*:**

What makes this algorithm different from TaskEntry case algorithm is just the update of total budgets before resource re-allocation. In fact, when a task of index '*idx*' exits the total budget and minimum budget decrease. And if task is adaptive, total adaptive budget decreases too. After budgets re-allocation, these total budgets are updated just like the previous case.

3) *Bandwidth variation (availableBw)*

Algorithm:

```
Begin
    Bw ← availableBw
    Calculate the new totMinBw_Bgt using the new Bw
    If (totMinBw_Bgt >1) then
        Bw ← total minimum Bw

    For each task
        minDiff ← $\frac{minBw}{Bw}$ - old minBw_Bgt
        minBw_Bgt ← $\frac{minBw}{Bw}$
        Bw_bgt ← Bw_bgt + minDiff
        totBw_Bgt ← totBw_Bgt + minDiff
        totMinBw_Bgt ← totMinBw_Bgt + minDiff
    EndFor

    For each adaptive task
        Bw_bgt ← Bw_bgt - $\frac{adaptiveBw\_Bgt}{totAdaptiveBw\_Bgt}$ * (totBw_Bgt - 1)
        AdaptiveBw_Bgt ← Bw_bgt – minBw_Bgt
        maxBw ← Bw_bgt * Bw
    EndFor

    totBw_Bgt ← 1
    totAdaptiveBw_Bgt ← 1 – totMinBw_Bgt
End
```

***Description*:**

This scenario differs from the two previous ones by the fact that the bandwidth resource has changed. That's why we have to update the total bandwidth budgets differently. First, we calculate the new total minimum requested budget using the new available bandwidth. We do this to test if the bandwidth decrease has an effect on the user minimum budgets preferences. If it has, we suppose that the available bandwidth is equal to the total minimum bandwidth of the task set in order to minimize the damage.

The total budget is then changed by updating the budgets of non-adaptive tasks and the minimum budgets of adaptive ones which are determined by the ratio of the task minimum bit rate (for non-adaptive tasks minBw equals maxBw) with the new available bandwidth. The total adaptive budget does not change since what is added to the total budget and the minimum total budget are changed equally.

If the bandwidth has decreased, the total bandwidth budget increases, i.e. exceeds 1, and then the budgets of adaptive tasks decrease, and vice versa. In both cases, the adaptive bandwidth budgets are re-allocated using the same expression as the previous algorithms.

Fairness goals

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

6. Generation of application configurations

Our adaptation framework is based on a configuration table whose records represent the generated QoS levels. The construction of the configuration table is a composite process of several steps described in the following sections.

6.1 Design of an application configuration

The first step in the table construction process is to determine the set of configuration parameters of the target application (i.e. whose values can be changed dynamically), the QoS parameters and their value domains.

Multimedia systems are characterized by QoS parameters which represent performance attributes. Some relevant QoS parameters are: video frame rate, level of resolution and number of colors (application layer); processor cycles and task periods and deadlines (system layer); display resolution, power processing and memory capacity (devices layer); and bit rate, network delay and packet loss rate (network layer).

QoS parameters can suffer uncontrolled variations in their values during delivery of continuous media data mainly due to network congestion and system load. Such variations affect the perceived quality and, consequently, the user's satisfaction.

We design a QoS level as follow. Each QoS level is characterized by a set of n configuration parameters { p1, p2, …pn } and p QoS parameters { QoS1, QoS2, …QoSp} which are application dependent. A QoS level, which is an entry of the configuration table, is then a multidimensional variable of a (n + p)-tuples representing a combination of values of the n configuration parameters.

6.2 Automatic configurations generation

For the purpose of automation of the configurations generation, we use the Multi-Objective-Particle Swarm Optimization (MO-PSO) algorithm [33]. The parameters previously selected are introduced in the MO-PSO algorithm to generate QoS scalable parameter sets which range from < highest quality/highest complexity > to < lowest quality/lowest complexity>.

6.2.1 Algorithm choice

Many real-world optimization problems have multiple objectives which interact and even possibly have conflicts with each other. In general, a multi-objective minimization problem with m decision variables (parameters) and n objectives can be stated by (E9) as follows:

$$\begin{cases} \textit{Minimize } y = f(x) = (f_1(x),\dots, f_n(x)) & \text{(E 9)} \\ \textit{Where} \quad x = (x_1,\dots, x_m) \in X \\ \qquad\qquad y = (y_1,\dots, y_n) \in Y \end{cases}$$

The MO-PSO algorithm has been proposed for such optimization. Many reasons have been behind our choice of the Particle swarm optimization. First, it is very widely used across a wide range of applications. Some tests in the literature [34] have found its implementation to be effective with several kinds of problems and it has been found to be very effective in a wide variety of applications, being able to produce very good results at a very low computational cost. In addition, it comprises a simple concept, i.e. requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory requirements and speed compared to other evolutionary algorithms. Also, many source codes of PSO are available in the Internet.

6.2.2 MO-PSO overview

J. Kennedy and R. C. Eberhart [35] originally proposed the PSO algorithm for optimization. PSO is a population-based search algorithm based on the simulation of the social behavior of birds within a flock [33].

6.2.2.1 Common terminology

For good understanding, we start by providing some definitions of several technical terms commonly used:

- Swarm: Population of the algorithm.
- Particle: Member (individual) of the swarm. Each particle represents a potential solution to the problem being solved. The position of a particle is determined by the solution it currently represents.
- *pbest* (*personal best*): Personal best position of a given particle, so far. That is, the position of the particle that has provided the greatest success (measured in terms of a scalar value analogous to the fitness adopted in evolutionary algorithms).
- *lbest* (*local best*): Position of the best particle member of the neighborhood of a given particle.
- *gbest* (*global best*): Position of the best particle of the entire swarm.
- Leader: Particle that is used to guide another particle towards better regions of the search space.
- Velocity (vector): This vector drives the optimization process, that is, it determines the direction in which a particle needs to “fly” (move), in order to improve its current position.

6.2.2.2 Algorithm

```
Begin

        Initialize swarm
        Initialize leaders in an external archive
        Quality (leaders)
        $g = 0$
        While g < gmax
                For each particle
                        Select leader
                        Update Position (Flight)
                        Mutation
                        Evaluation
                        Update pbest
                EndFor
                Update leaders in the external archive
                Quality(leaders)
                $g$++
        EndWhile
                Report results in the external archive
End
```

Figure 11. *Pseudo-code of a general MO-PSO algorithm*

Figure 11 shows the way in which a general MOPSO algorithm works. First, the swarm is initialized. This initialization includes both positions and velocities. Then, a set of leaders is also initialized with the non-dominated particles from the swarm. The set of leaders is usually stored in an external archive. Later on, some sort of quality measure is calculated for all the leaders in order to select (usually) one leader for each particle of the swarm. At each generation, for each particle, a leader is selected and the flight is performed, i.e. the particle flies through the search space through updating its position.

Most of the existing MOPSOs apply some sort of mutation operator after performing the flight. Then, the particle is evaluated and its corresponding pbest is updated. A new particle replaces its pbest particle usually when this particle is dominated or if both are incomparable (i.e., they are both non-dominated with respect to each other). After all the particles have been updated, the set of leaders is updated, too. Finally, the quality measure of the set of leaders is recalculated. This process is repeated for a certain (usually fixed) number of iterations.

6.2.2.3 Output

The solution set of a problem with multiple objectives does not consist of a single solution (as in global optimization). Instead, in multi-objective optimization, we aim to find a set of different solutions (the so-called Pareto optimal set). The optimal trade-off solutions among objectives constitute the Pareto front [36].

In general, when solving a multi-objective problem, three are the main goals to achieve [33]:

- Maximize the number of elements of the Pareto optimal set found.
- Minimize the distance of the Pareto front produced by our algorithm with respect to the true (global) Pareto front (assuming we know its location).
- Maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible.

6.2.3 Configuration of the MO-PSO

Four main steps have to be performed in order to interface the MO-PSO with the target application:

- Define the population variables (or particles) which are the application configuration and QoS parameters previously determined.
- For each particle *pi* set the domain of values [$p_i$min, $p_i$max].
- Define the objective functions: number of objectives, optimization type (minimization or maximization), and evaluation functions.
- Put constraints, if any, on particles to determine their feasibility when added to the solutions archive.

Since different combinations of configuration parameter values can have similar requirements of bit rate or processing time, or represent similar qualities, we need to put some assumptions on the generated QoS levels in order to discard undesirable ones. For example, let’s consider two QoS levels Li and Lj of a video application, with two QoS parameters: QoS for video output quality and B for its bit rate. If QoSi > QoSj, but Bi < Bj , the table should held only Li. That is, it should be kept only QoS levels with quality degrees as highest as possible and requiring less bandwidth. The same assumption is applicable for the processing time. These assumptions are a priori considered by the MO-PSO algorithm since it provides a Pareto front composed of the non-dominated optimal trade-off solutions.

At the end of this step, we obtain a set of application configurations. It will be used in the online adaptation step when we have to select the best configuration that satisfies the resources constraints.

7. Application adaptor

The application adaptor design is the final step in our work flow. We focus on two essential points: the best configuration selection and the adaptation stability.

7.1 Configuration selection

First, a selector picks the best configuration from a pre-established configuration table that:

i) maximizes the quality of service,

ii) best meets the received constraints. It considers ensuring the coordination between coarse and fine time granularity adaptations, i.e. the first respects the resource allocations made by the last, and vice versa.

After selecting the best configuration, the application has to be reconfigured dynamically.

Our adaptive technique offers a flexible design suitable for accommodating new applications, without loss of generality and configurability. To fully support a new application, an appropriate application-specific adaptor has to be written and incorporated into our adaptive system.

7.2 Adaptation stability

Our application-aware adaptation technique needs to be aware not only of the application-specific adaptation objectives but also of the system-wide requirements such as stability and fairness among concurrent applications. We introduce hereafter these system-wide properties which then need to be thoroughly studied.

When designing an adaptive system, it is obvious to think about the adaptation stability it shows. To avoid all-time adaptation which leads to unstable system, threshold adaptations can be used. When variations in resource availability exceed threshold, adaptation must be activated. This is an important solution that we may consider in order to guarantee the stability of the system. Furthermore, researchers in the literature, such in [37], have dealt with adaptation stability analysis and have defined it as to be related to two categories of the system dynamics:

1) Statistical multiplexing. In an environment of concurrent tasks sharing limited resources, when the number of active tasks is fixed, system resources allocated to each task should settle down to an equilibrium value in a definite period of time. This also implies that, if a new task becomes active, existing active tasks will adjust their resource usage so that after a brief transient period, the system settles down to a new equilibrium.

2) Disturbances. Stability also implies that adaptation activities do not suffer from oscillations despite variations in resource availability due to unpredictable and physical causes, such as network congestion or volatile wireless connection. Oscillations are undesirable because they affect the user-perceptible quality and attempt to occupy so many resources because of an excessive amount of adaptation (important overhead).

With regard to the first point, our system is able to converge to an equilibrium value of resources allocated to each task. This is possible thanks to the coordination between the resource controllers and the GRM which permits a prompt adjustment of resource budgets in response to a resource variation event.

As for the second point, adaptation approaches must deal with the considerable uncertainty related to congestion determination, as the network load oscillates very suddenly and drastically. Since we manage soft real time applications that accept some resource exceeding, a trivial solution to reduce adaptation oscillations is the use of tolerance intervals. Other solutions have been proposed in the literature to deal with resource control vagueness and uncertainty. For instance, in [37] and [38] authors have adopted soft computing techniques, such as fuzzy control, artificial neural networks and neuro-fuzzy control. They have proposed adaptation approaches specifically based on fuzzy controllers and have demonstrated them to be promising approaches in assisting QoS adaptation in distributed multimedia systems. However, we have to note that these approaches have been dedicated to distributed Client/Server systems which differ from embedded systems at multiple levels (e.g. resource limitation, external environment variation). Moreover, fuzzy controllers have a significant HW integration cost. So, much effort must be done to efficiently integrate them in an embedded system. Thus, we plan in future works to integrate soft computing techniques in our adaptation approach and study their effectiveness for embedded systems (i.e. overhead).

8. Conclusion

In this chapter, we developed our multi-constraints adaptation technique which enables embedded systems to adapt to changes in system load and network resource at different time granularities. We presented our adaptation as a tradeoff between quality of service, available network bandwidth, and real time constraint. We gave a formal description of the system and formulation of the problem.

Thereafter, we detailed the different design steps that we followed to build the adaptation technique. It consists of an offline characterization step for QoS levels generation, and an in-line adaptation step that changes the application software configuration so that the output quality is maximized while the resource constraints are met (CPU and bandwidth allocation). We proposed a novel method for an automatic generation of QoS-levels dedicated for applications holding a large set of configuration parameters. It is based on the Multi-Objective Particle Swarm Optimization algorithm.

In the next chapter, we detail the implementation of the proposed technique and present the obtained results.

Chapter 3

Experimentation and validation

1. Introduction

Our target systems are embedded systems running multimedia applications. As a case study to apply our adaptation mechanism, we consider a complex multimedia application which is the H264/AVC video coding.

In this last chapter, we give an overview of the H264/AVC video coding system, and the additional features it brings.

We can note that the operational control of the encoder is a key problem in video compression. In fact, to encode a source video, many coding parameters such as profile, level, QP, ME search algorithm, and macroblock modes have to be determined. The chosen values of these parameters determine the coding efficiency of the produced bit-stream of a given encoder. Therefore, as a first step to generate the application configurations, we realize the study of the application and the selection of possibly tuned configuration parameters as well as the set of QoS parameters. Then we present the implementation and tests of the adaptation technique using the video coding application.

2. Target system

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. We are particularly interested in streaming video application. Figure 12 presents the general context of our target system.

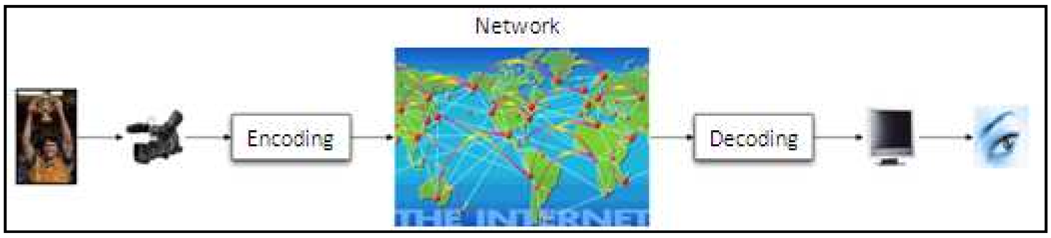


Figure 12. *Target system structure*

In the Césame project, Xilinx FPGA is used as the embedded system technology. We plan to use two Xilinx Virtex-5 ML 507 FPGA boards connected using Ethernet port to exchange a direct video. Before implementation on the FPGA board, we first validate our technique using a PC. Figure 13 illustrates the ML 507 board.

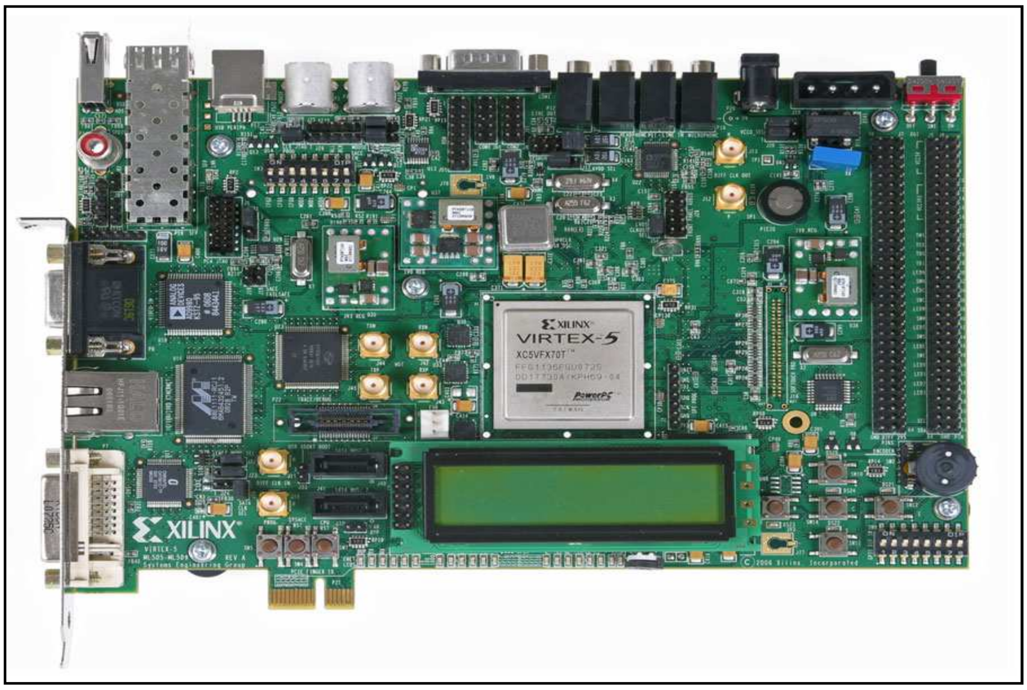


Figure 13. *ML 507 FPGA board*

This board is very rich in multimedia and network peripherals. It is essentially composed of:

- DDR2 memory (256 Mo)
- USB (2) connectors
- PS/2 (2) – keyboard, mouse
- RJ-45 – 10/100/1000 networking
- RS-232 (Male) – serial port
- Video VGA Input
- Video (DVI/VGA) output
- Audio In (2) – line, microphone
- Audio Out (2) – line, amp, SPDIF, piezo speaker
- PCI Express edge connector

Targeted markets of the ML 507 FPGA board are industry, telecom, medicine, military and aerospace. Its targeted applications are, among others, data transmission and manipulation, serial connectivity, digital video, bus interface and high speed design.

3. H264/AVC: a case study application

In this section, we present the video coding technique as an example of multimedia application. We start by describing the generalized block diagram of a video coding system. Then, we describe the new coding features that have appeared with the H264/AVC video coding standard. More detailed description can be found in [39] [40] [41] [42].

3.1 H264/AVC vs. previous standards

The standard H.264 or MPEG-4 part 10 Advanced Video Coding (AVC) is one of the most important developments in video coding in the last few years that has been defined jointly by the Joint Video Team (JVT) of the ITU and the ISO/IEC. It is the latest standard in a sequence of the video coding standards H.261 (1990), MPEG-1 Video (1993), MPEG-2 Video (1994), H.263 (1995, 1997), MPEG-4 Visual or part 2 (1998). These previous standards reflect the technological progress in video compression and the adaptation of video coding to different applications and networks [39].

Although these standards define similar coding algorithms, they contain features and enhancements that make them differ. The improvements made by H264 involve mainly the formation of the prediction signal; the block sizes used for transform coding, and the entropy coding methods. In [40] the performance of the various standards is compared by means of PSNR and subjective testing results. The results indicate that H.264/AVC compliant encoders typically achieve essentially the same reproduction quality as encoders that are compliant with the previous standards while typically requiring 60% or less of the bit-rate.

3.2 H264/AVC overview

3.2.1 Block diagram of video encoder

All Standardized ITU-T and ISO/IEC JTC1 video coding techniques like MPEG-1, 2, 4, H.263, H.264/AVC, have in common that they are based on a block-based hybrid video coding. Figure 14 shows the generalized block diagram of this hybrid video encoder [39].

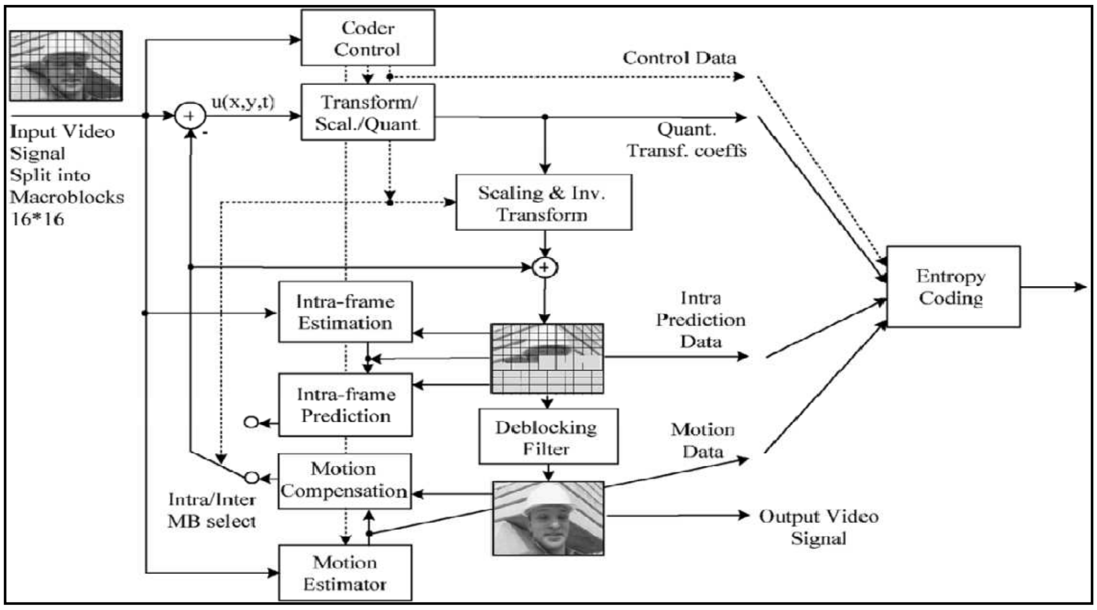


Figure 14. *Block diagram of hybrid video encoder (especially for H.264/AVC)*

Each image of the input video signal is partitioned into fixed-size macroblocks of 16x16 samples. These macroblocks are coded in Intra or Inter mode.

In Inter mode, a macroblock is predicted using motion compensation, exploiting temporal statistical dependencies between different pictures. For motion compensated prediction, a displacement vector (motion vector) is estimated and transmitted for each block (motion data) that refers to the corresponding position of its image signal in an already transmitted reference image stored in memory.

In Intra mode, former standards set the prediction signal to zero such that the image can be coded without reference to previously sent information.

The prediction residual, which is the difference between the original and the predicted block, is then further compressed using a transform to remove spatial correlation inside the transform block before it is quantized and entropy coded.

In order to reconstruct the same image on the decoder side, the quantized coefficients are inverse-transformed and added to the prediction signal. The result is the reconstructed macroblock that is also available at the decoder side. This macroblock is stored in a memory.

3.2.2 Picture coding modes

In a video codec, an individual frame may be encoded using one of three modes: I, P or B (c.f.Figure 15).

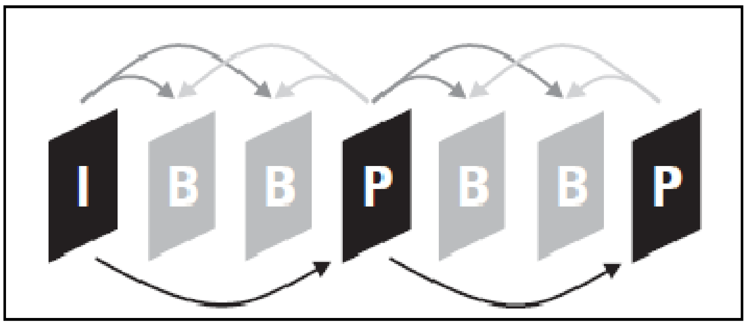


Figure 15. *Illustration of Inter-Frame Prediction in I, P and B Frames*

In INTRA coded pictures or I-pictures, all macroblocks are coded independently without referring to other pictures in the video sequence (no motion compensation).

P and B-pictures are predictive-coded pictures where typically one of a variety of INTER coding modes can be chosen to encode each macroblock. P-frames are coded using MC with a previous frame as reference (forward prediction). B frames or bi-directional predicted frames are predicted from both past frames as well as frames slated to appear after the current frame.

3.2.3 Video bitstream hierarchy

A video bitstream is usually arranged in layers, each corresponds to a hierarchy, as is shown in Figure 16 [43].

- Video sequence: it is the highest layer. It consists of sequence header, the number of group of pictures (GOP), and end-of-sequence code.
- Group of pictures: it consists of a series of one or more pictures intended to allow random access into the sequence. A GOP always starts with an I-picture.
- Picture: The picture is the primary coding unit of the video sequence which consists of 3 rectangular matrices representing one luminance (Y) and two chrominance (Cb and Cr) components.

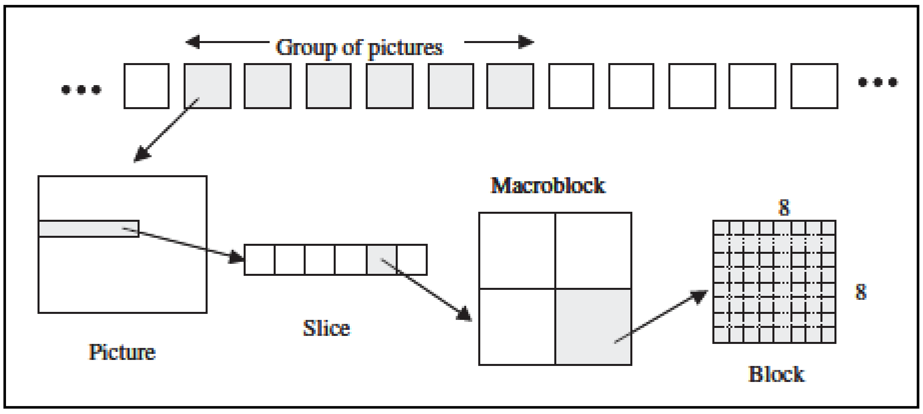


Figure 16. *Video bitstream hierarchy*

- Slice: A slice is the layer for Intra frame addressing and (re)synchronization. It consists of one or more contiguous macroblocks. Slices should have the similar amount of data.

- Macroblock (MB): Each macroblock consists of the three components Y, Cr and Cb. Due to the fact that the human eye system is less sensitive to the chrominance than to the luminance, the chrominance signals are both sub-sampled by a factor of 2 in horizontal and vertical direction. Therefore, a typical macroblock consists of one block of 16 by 16 picture elements for the luminance component and of one or two blocks of 8 by 8 picture elements for the color components depending on whether it is 4:2:0 or 4:2:2 sub-sampled. The MB is the basic unit for motion prediction and compensation.

- Blocks: A block is an 8×8 pixel section of luminance or chrominance components. It is the basic unit for DCT based Intra frame coding.

3.2.4 Profiles and levels

In order to manage the large number of coding tools included in standards and the broad range of formats and bit-rates supported, the concept of profiles and levels is typically employed to define a set of conformance points, each targeting a specific class of applications. These conformance points are designed to facilitate interoperability between various applications of the standard that have similar functional requirements. A profile defines a set of coding tools or algorithms that can be used in generating a compliant bit-stream, whereas a level places constraints on certain key parameters of the bit-stream, such as the picture resolution and bit-rate.

3.2.5 *H264/AVC new features*

In the following, we give a description of the H264 new features which are behind its great performance enhancement.

While H.264 uses the same general coding techniques as previous standards, it has many new features that distinguish it from previous standards [41], that combined improve coding efficiency. The main differences are summarized in the encoder block diagram in Figure 17 and are also briefly described below.

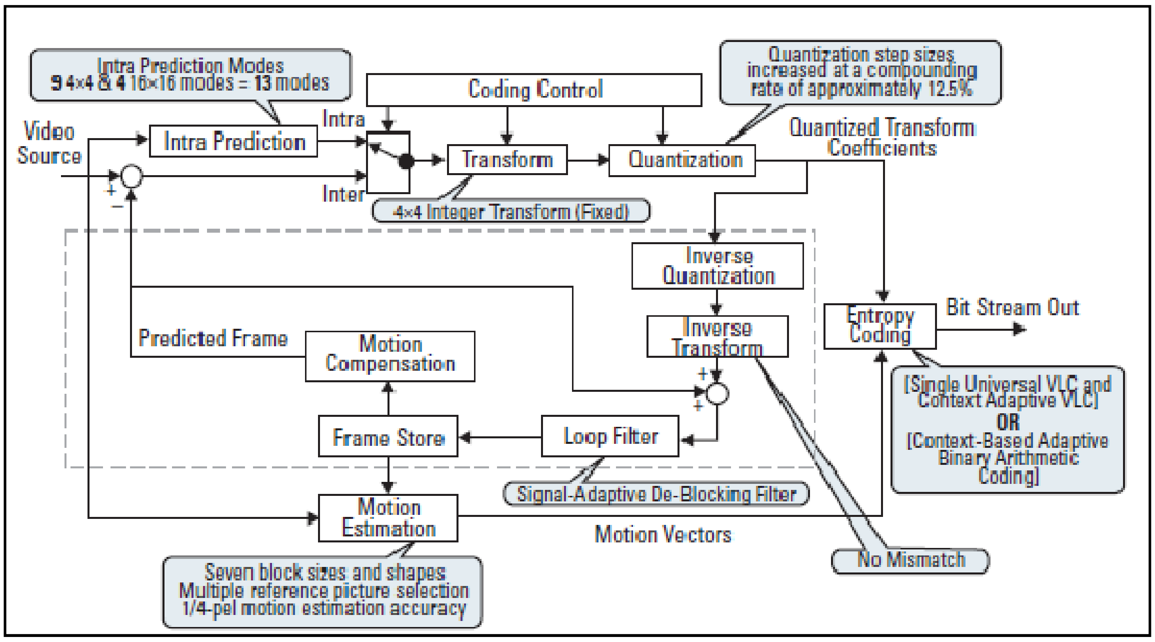

Intra Prediction Modes
9 4×4 & 4 16×16 modes = 13 modes
Coding Control
Quantization step sizes increased at a compounding rate of approximately 12.5%
Video Source
Intra Prediction
Intra
Inter
Transform
Quantization
Quantized Transform Coefficients
4×4 Integer Transform (Fixed)
Inverse Quantization
Inverse Transform
Entropy Coding
Bit Stream Out
[Single Universal VLC and Context Adaptive VLC]
OR
[Context-Based Adaptive Binary Arithmetic Coding]
Predicted Frame
Motion Compensation
Frame Store
Loop Filter
Signal-Adaptive De-Blocking Filter
Motion Estimation
Motion Vectors
No Mismatch
Seven block sizes and shapes
Multiple reference picture selection
1/4-pel motion estimation accuracy

Figure 17. *H264 block diagram, new features*

3.2.5.1 *Intra picture prediction*

H.264 uses spatial domain Intra prediction to predict the pixels in an Intra-MB from the neighboring pixels in adjacent blocks. The prediction residual along with the prediction modes is coded rather than actual pixels in the block. This results in a significant improvement in intra coding efficiency.

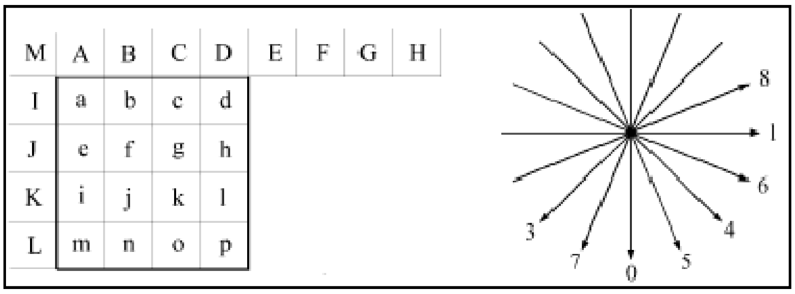


Figure 18. *Intra 4_4 prediction: samples and directions*

In all slice-coding types, two primary types of Intra coding are supported: Intra 4x4 and Intra 16x16 prediction. Chroma Intra prediction is the same in both cases. A third type of Intra coding, called I PCM, is also provided for use in unusual situations.

The Intra 4x4 mode is based on predicting each 4x4 luma block separately and is well suited for coding of parts of a picture with significant detail. The Intra 16x16 mode, on the other hand, does prediction and residual coding on the entire 16x16 Luma block and is more suited for coding very smooth areas of a picture. In addition to these two types of luma prediction, a separate chroma prediction is conducted.

In contrast to previous video coding standards (especially H.263 and MPEG-4 Visual), where Intra prediction has been conducted in the transform domain, Intra prediction in H.264/AVC is always conducted in the spatial domain, by referring to neighboring samples of previously decoded blocks that are to the left and/or above the block to be predicted. In Intra 4x4 mode, each 4x4 luma block is predicted from spatially neighboring samples as illustrated on the left-hand side of Figure 18. The 16 samples of the 4x4 block, marked a–p, are predicted using position-specific linear combinations of previously decoded samples, marked A–M, from adjacent blocks. The encoder can select either “DC” prediction (called mode 2, where an average value is used to predict the entire block) or one of eight directional prediction types illustrated on the right-hand side of Figure 18. The directional modes are designed to model object edges at various angles.

In Intra 16x16 mode, the whole 16 16 Luma component of the macroblock is predicted at once, and only four prediction modes are supported: vertical, horizontal, DC, and plane. The first three are similar to the modes in Intra 4x4 prediction except for increasing the number of samples to reflect the larger block size. Plane prediction uses position-specific linear combinations that effectively model the predicted block as a plane with an approximate fit for the horizontal and vertical variation along the block edges.

The chroma samples of an Intra macroblock are predicted using similar prediction techniques as for the luma component in Intra 16x16 macroblocks.

3.2.5.2 Inter picture prediction

Inter-frame coding in H.264 leverages most of the key features in earlier standards and adds both flexibility and functionality, including multiple options for block sizes for motion compensation, quarter-pel motion compensation, multiple-reference frames, generalized bi-directional prediction and adaptive loop de-blocking.

- *Variable Vector Block Size*

Motion compensation can be performed using a number of different block sizes. Individual motion vectors can be transmitted for blocks as small as 4×4, so up to 32 motion vectors may be transmitted for a single MB in the case of bi-directional prediction. P macroblocks can be partitioned into smaller regions for motion-compensated prediction (MCP) with luma block sizes of 16x16, 16x8, 8x16, and 8x8 samples. When 8x8 macroblock partitioning is chosen, an additional syntax element is transmitted for each 8x8 partition, which specifies whether the 8x8 partition is further partitioned into smaller regions of 8x4, 4x8, or 4x4 luma samples and corresponding chroma samples (see Figure 19).

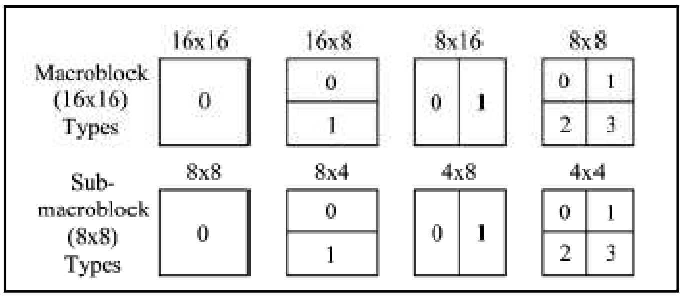


Figure 19. *Segmentations of Macroblock and 8x8 partitions for MC*

Smaller block sizes improve the ability to handle fine motion detail and results in better subjective quality including the absence of large blocking artifacts.

The prediction signal for each predictive-coded luma block is obtained by MC, which is specified by a translational MV and a picture reference index. The syntax allows MVs to point over picture boundaries. The MV values are differentially coded using either median or directional prediction from neighboring blocks. No MV value prediction (or any other form of prediction) takes place across slice boundaries.

- *Multiple reference frames*

Indeed, the syntax supports multipicture MCP. Up to 16 different reference frames can be used for inter-picture coding. Previously decoded pictures are stored in a decoded picture buffer (DPB) as directed by the encoder, and a DPB reference index is associated with each motion-compensated 16x16, 16x8, 8x16, or 8x8 luma block. MCP for smaller regions than 8x8 uses the same reference index for predicting all blocks in an 8x8 region.

Providing multiple reference frames results in better subjective video quality and more efficient coding. It can also help make the H.264 bitstream more error resilient. However, this feature leads to increased memory requirement for both the encoder and the decoder since multiple reference frames must be maintained in memory.

- *Fractional-sample-accuracy*

Motion compensation is improved by allowing half-pixel and quarter-pixel motion vector resolution. This refers to the use of spatial displacement MV values that have more than integer precision, thus requiring the use of interpolation when performing MCP.

- *Adaptive Loop De-blocking Filter*

H.264 uses an adaptive de-blocking filter that operates on the horizontal and vertical block edges within the prediction loop to remove artifacts caused by block prediction errors. The filtering is generally based on 4×4 block boundaries, in which up to three pixels on either side of the boundary may be updated using a 4-tap filter.

- *Integer Transform*

Previous standards that use DCT had to define rounding-error tolerances for fixed-point implementations of the inverse transform. Drift caused by mismatches in the IDCT precision between the encoder and decoder were a source of quality loss. H.264 gets around the problem by using an integer 4×4 spatial transform, which is an approximation of the DCT. The small 4×4 shape also helps reduce blocking and ringing artifacts.

- *Quantization and Transform Coefficient Scanning*

Transform coefficients are quantized using scalar quantization with no widened dead zone. Different quantization step sizes can be chosen for each MB, similar to prior standards, but the step sizes are increased at a compounding rate of approximately 12.5%, rather than by a constant increment. Also, finer quantization step sizes are used for the chrominance component, especially when the luminance coefficients are coarsely quantized.

- *Entropy Coding*

Unlike previous standards that offered a number of static VLC tables depending on the type of data under consideration, H.264 uses a Context-Adaptive VLC for the transform coefficients and a single universal VLC approach for all the other symbols. The main profile also supports a new Context-Adaptive Binary Arithmetic Coder (CABAC). The CAVLC is superior to previous VLC implementations but without the full cost of CABAC.

CABAC uses a probability model to encode and decode the syntax elements such as transform coefficients and motion vectors. To increase the coding efficiency of arithmetic coding, the underlying probability model is adapted to the changing statistics within a video frame through a process called context modeling. Context modeling provides estimates of conditional probabilities of the coding symbols. Utilizing suitable context models, the given inter-symbol redundancy can be exploited by switching between different probability models, according to already coded symbols in the neighborhood of the current symbol. Each syntax element maintains a different model (for example, motion vectors and transform coefficients have different models). CABAC can provide up to about 10% bit rate improvement over CAVLC.

- *Weighted Prediction*

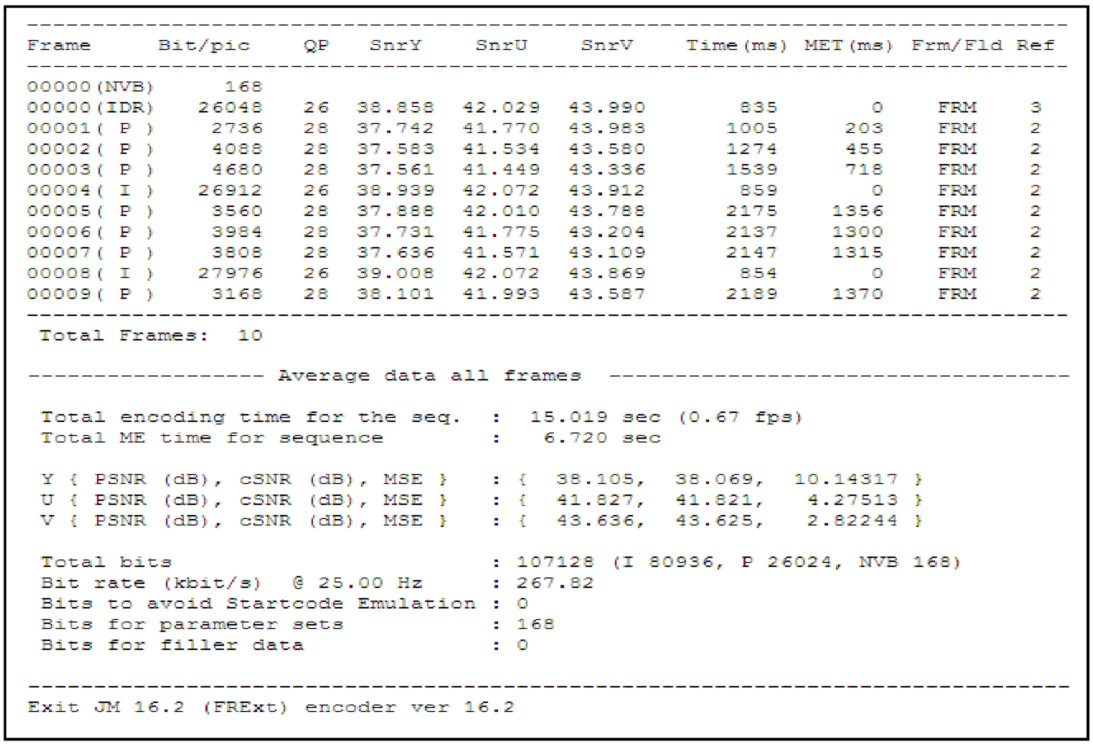
It forms the prediction for bi-directionally interpolated macroblocks by using the weighted sum of forward and backward predictions, which leads to higher coding efficiency when scene changes fade.

3.3 *H264/AVC software*

As an implementation of the H264/AVC, we used the open source JM software. The software package contains a Visual Studio .NET workspace including a H.264/AVC reference encoder “lencod” and a H.264/AVC reference decoder “ldecod” [44]. This software can be found at [45].

To run the JM encoder, we need the lencod.exe file, the configuration file named “encoder.cfg” and the input video sequence (YUV format). The current software only supports concatenated input sources (i.e. all components and frames should be included in a single file).

The JM software generates output rate/distortion statistics for every coded frame, such as number of bits, PSNR, frame encoding time and ME time.



```
-------------------------------------------------------------------------------
Frame      Bit/pic    QP   SnrY    SnrU    SnrV    Time(ms) MET(ms) Frm/Fld Ref
-------------------------------------------------------------------------------
00000(NVB)      168
00000(IDR)    26048   26  38.858  42.029  43.990       835       0     FRM    3
00001( P )     2736   28  37.742  41.770  43.983      1005     203     FRM    2
00002( P )     4088   28  37.583  41.534  43.580      1274     455     FRM    2
00003( P )     4680   28  37.561  41.449  43.336      1539     718     FRM    2
00004( I )    26912   26  38.939  42.072  43.912       859       0     FRM    2
00005( P )     3560   28  37.888  42.010  43.788      2175    1356     FRM    2
00006( P )     3984   28  37.731  41.775  43.204      2137    1300     FRM    2
00007( P )     3808   28  37.636  41.571  43.109      2147    1315     FRM    2
00008( I )    27976   26  39.008  42.072  43.869       854       0     FRM    2
00009( P )     3168   28  38.101  41.993  43.587      2189    1370     FRM    2
-------------------------------------------------------------------------------
 Total Frames:  10

------------------ Average data all frames  -----------------------------------

 Total encoding time for the seq.  :  15.019 sec (0.67 fps)
 Total ME time for sequence        :   6.720 sec

 Y { PSNR (dB), cSNR (dB), MSE }   : {  38.105,  38.069,  10.14317 }
 U { PSNR (dB), cSNR (dB), MSE }   : {  41.827,  41.821,   4.27513 }
 V { PSNR (dB), cSNR (dB), MSE }   : {  43.636,  43.625,   2.82244 }

 Total bits                        : 107128 (I 80936, P 26024, NVB 168)
 Bit rate (kbit/s)  @ 25.00 Hz     : 267.82
 Bits to avoid Startcode Emulation : 0
 Bits for parameter sets           : 168
 Bits for filler data              : 0

-------------------------------------------------------------------------------
Exit JM 16.2 (FRExt) encoder ver 16.2
```

Figure 20. *Example of JM output rate/distortion statistics*

At the end of the encoding process the encoder output presents also cumulative results such as total encoding time, total ME time, average PSNR and bit rate (c.f. Figure 20). We need these statistics to generate our application configurations. Another cumulative statistic that we need to add is the average encoding time per frame.

4. Implementation and test of the offline characterization step

Configurations generation requires the determination of both QoS and configuration parameters set, then the interfacing of the application with the MO-PSO algorithm. These steps are described in the next sub-sections.

4.1 Configuration parameters selection

In order to select the configuration parameters set for the H264/AVC encoder, we analyzed the complexity of key functional blocks of this latter to determine the most important parameters having great effect on video quality and encoding complexity. We have focused on some general encoder control parameters such as quantization parameter (QP), intra-period, search range and number of reference frames. We have also focused in particular on the control parameters of the ME block, which is a key enhancement module incorporated in the H264/AVC standard, but is also heavy on processor cycle consumption.

[22] gives a quantization of the complexity of ME bloc. It defines it as a function of MB size NXN, integer pixel and fraction pixel full search ME computations for a MB, search range, number of reference blocks, the ME metric computation cost, and the interpolation cost for a MB in fraction pel ME.

We have also synthesized from other works [46] [39] [41] a complexity analysis of key parameters in ME block which is summarized in table 1:

- Number of reference frames (Nref) (up to 16),
- Search range (SR)
- ME algorithm (SearchMode)
- Bloc size (taille de Bloc de 16x16, 16×8, 8×16, 8×8, 8×4, 4×8 et 4x4)
- pixel resolution (full, half and quarter pixel accuracy)

Table 1. *Complexity analysis of key parameters in ME block*

H264 encoding tools	**Complexity**	
	Access frequency	Quality
Variable Block Sizes	Linear increase: more than 2.5% complexity increase for each additional mode while the corresponding compression gain saturates	bit rate reduction between 4 and 20% (for the same quality)
Displacement vector resolution	- About 10% access frequency and processing time reduction when searching for MV only at ½ pixel positions instead of ¼ - Up to 30% coding efficiency increase when using 1/4 pel MV except for very low bit rates.	–
RD-Lagrangian optimisation	Increase in the order of 120%	Up to 9% bit rate reduction Up to 0.35dB PSNR improvement.
Search Range	Up to approximately 60 times when increasing search size and Nref	Minimal impact on PSNR and bit rate performances
Multiple reference frames	Linear model increase: 25% complexity increase for each added frame	- A negligible bit rate gain (less than 2%) for low and medium bit rates, - More significant savings can be achieved for high bit rate sequences (up to 14%)

From the previous study, we find that many possibilities exist for the selection of adaptive QoS parameters for the application H264/AVC encoder. We have tried different parameter sets including essentially the following parameters:

- **IntraPeriod**, Max period of I-coded frames (non IDR) in the encoded sequence. A value of 0 (default) implies that only the first frame will be coded as intra.
- **QPISlice**,which **s**ets quantization parameter (QP) value for I slices. Allowable values are in the range of 0 to 51. Default value is 24.
- **SearchMode,** the ME algorithm whose options are given inTable 2.

Table 2. *ME algorithm options*

Option	Search algorithm
-1	Full Search
0	Fast Full Search (default)
1	UMHexagon Search
2	Simplified UMHexagon Search
3	Enhanced Predictive Zonal Search (EPZS)

- **Inter Mode Prediction Control such as** PSliceSearch8x4, PSliceSearch4x8, PSliceSearch4x4, which control which inter prediction modes could be used for encoding purposes. For example, PSliceSearch4x4 enables 4x4 Inter Prediction & Motion Compensation in P Slices. The default value is 1 (enabled).

4.2 QoS parameters selection

Concerning the QoS application parameters, we consider a set of three parameters *{ QoS, Br, Tex}* which are respectively a video quality metric, the average encoding time/frame and the bit rate. The choice of the two latter parameters is explained by the fact that our system is CPU and bandwidth constrained. As for the former, the video quality metrics represent a large field of research that is still under development. Different video quality metrics have been proposed in the literature. There are usually three ways of measuring the coding quality or fidelity [43]:

(1) Subjective Assessment, where human observers evaluate the reconstructed video focusing on aesthetic acceptability;

(2) Objective Assessment. That is, to numerically compare the pixel values before and after coding;

(3) Perceptual Metrics. These are computational algorithms that could accurately predict the resulting scores of the subjective assessment. That is by comparison with the human perceptual mechanism.

4.2.1 Subjective Assessment

A group of human subjects is invited to judge the quality of video sequences under defined conditions. The main subjective quality methods are Degradation Category Rating (DCR), Pair Comparison (PC) and Absolute Category Rating (ACR). The human subjects are shown two sequences (original and processed) and are asked to assess the overall quality of the processed sequence by scoring it on a scale (from 0 to 5 or 9) corresponding to their mental measure of the quality --this is termed Mean Observer Score (MOS).

It is noted that subjective assessment reflects the perceptual nature of the video signal under assessment. It is an extremely expensive and time consuming process. Indeed, it might not be consistent as human observers are easily affected by the circumstances which include their psychological condition, personal preferences, and viewing condition.

4.2.2 Objective Assessment

Subjective Video Analysis is only applicable for development and evaluation purposes; it does not lend itself to operational monitoring or production line testing. The need for Objective Video Quality Testing arose from the need for quantitative, repeatable video analysis.

The two most commonly used objective measurements used to assess the quality of the compressed video is the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). This comparison is typically done on a frame-by-frame basis. The MSE is the cumulative squared error between the compressed video frame and the original frame. The PSNR is a measure of the peak error. The MSE and PSNR are defined mathematically as:

$$\mathrm{MSE} = \frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=0}^{N-1}[I(x,y) - I'(x,y)]^2, \qquad \text{(E 10)}$$

$$\mathrm{PSNR} = 20 \cdot \log_{10}\frac{255}{\sqrt{\mathrm{MSE}}}, \qquad \text{(E 11)}$$

Where I(x, y) is the original video frame, I_(x, y) is the reconstructed frame and M and N are the dimensions of the video frame. A small value of MSE means lesser error, that is good quality, and this translates to a higher value of PSNR. A lower PSNR value represents poorer quality.

The advantage of objective assessment is that it is simple and straight forward. However, the results it delivers do not correlate well with perceived picture quality. For example, similar PSNR in the texture area and the smooth area of an image will exhibit different distortions when perceived by the human eye. The Human Vision System is less sensitive to errors in the texture area than the smooth area.

4.2.3 Perceptual Metrics

The aim of developing a perceptual metric for video quality is to find a better prediction for the human perception than the MSE/PSNR measure. There have been substantial research efforts in perceptual visual quality evaluation. These perceptual metrics have a better performance than the traditional MSE/PNSR measure for some specific applications; it is difficult to find a general perceptual metric suitable for all applications.

The Video Quality Experts Group (VQEG) has evaluated ten perceptual metrics which have been proposed by different researchers but none of them is statistically better than the PSNR.

We therefore choose the PSNR as a video quality metric. We plan in future works to define a new metric that is better than the PSNR.

4.3 MO-PSO/H264 interfacing

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

First, we give the definition of particles in initialize_pop() function. We define for each particle the composing variables and their value ranges that we have just mention in the previous section.

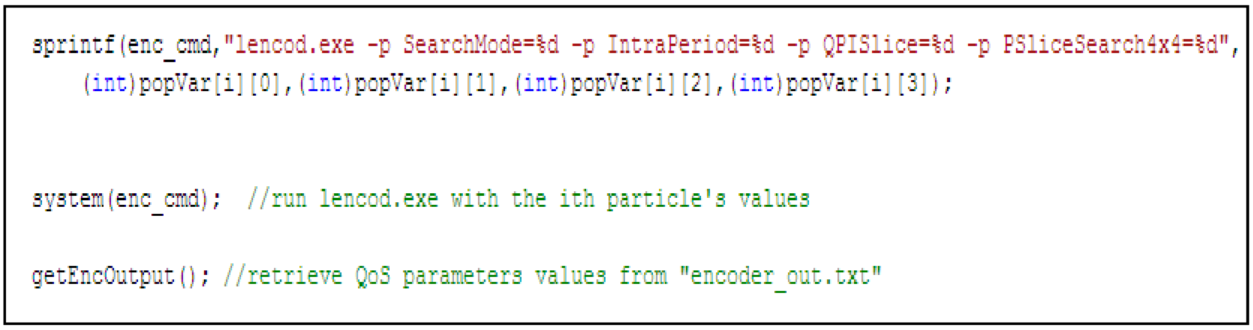
Second, since we need to optimize the trade-off between the output quality, the bit rate and the encoding time, we define three objective functions, *H264_Q()*, *H264_BR()* and *H264_Tex()*, which return respectively the QoS, Bit rate and processing time values for a given particle. Then, we mention the optimization type for each function (maximization or minimization). Initially, the MO-PSO code doesn’t support different optimization types. Thus, we were brought to make some modification of its source code to enable this property.

Thereafter, we need to specify how to evaluate particles in population in the *evaluate()* function. In our case, we execute “*lencod.exe*” for each particle *i* with the command line

-p <ParameterName = ParameterValue …ParameterName = ParameterValue >

Where *parameterName* is the name of the variables composing the particle, and *ParameterValue* is the current values of these variables in the particle *i*.

We give in Figure 21 an example of MO-PSO/H264 interfacing. We consider 4 variables per particle which are *SearchMode, IntraPeriod, QPISlice* and *PSliceSearch4x4*.



```
sprintf(enc_cmd,"lencod.exe -p SearchMode=%d -p IntraPeriod=%d -p QPISlice=%d -p PSliceSearch4x4=%d",
    (int)popVar[i][0],(int)popVar[i][1],(int)popVar[i][2],(int)popVar[i][3]);


system(enc_cmd);  //run lencod.exe with the ith particle's values

getEncOutput(); //retrieve QoS parameters values from "encoder_out.txt"
```

Figure 21. *Sample code for MO-PSO/H264 interfacing*

After running the encoder, the MO-PSO retrieves the output QoS parameters values which are the average encoding Time/frame, the Bit Rate and the average PSNR. These values are saved to a file, "encoder_out.txt", which we update at the end of every lencod.exe’s execution.

4.4 Assumptions on resources ranges

The last available version of the JM software is non-optimized so that the encoding time per frame is high. Thus, we make assumptions on the limits of system resources, namely the CPU time and network bandwidth, which are taken into account in the generated configurations. We used the MO_PSO algorithm to make an approximation of the minimum allowable encoding time per frame. We found a minimum encoding time of about 150ms/frame, i.e. a maximum of about 6.66 frame per second (fps). Therefore, in order to have a sufficient encoding time range to test our technique, we assume that a maximum of encoding time/ frame is equal to 500ms, i.e. a minimum of 2fps.

Concerning the bit rate, it ranges from tens to thousands of kbits. We assume that the maximum allowable bit rate is equal to 1.4Mb/s.

4.5 MO-PSO results

There are no explicit rules to tune the parameters of the MO-PSO. The best parameters tuning depends on the application considered for optimization, and are determined after multiple tests.

Let's consider the configuration of the MO-PSO software presented in Table 3. It contains the configuration parameters we tune.

Table 3. *Example of MO-PSO configuration*

Parameter	Value	Test application values
Population size	50	
Number of generations	50	
Archive size	50	
objective functions	3	H264_Tex(), H264_BR(), H264_Q()
Number of variables	7	SearchMode, IntraPeriod, QPISlice, QPPSlice PSliceSearch4x4, PSliceSearch4x8, PSliceSearch8x4,

The video type we consider for test is speaking-head videos with QCIF format (Quarter Common Intermediate Format, 176×144px) like "foreman.yuv" and "news.yuv". We perform multiple tests with different configurations of the MO-PSO. Each time we tune the number of particles, the number of generations and the archive size. We also change the configuration parameters of the H264/AVC encoder. We give hereafter the results of some examples of tests that help us to choose better configuration giving better solutions.

We first run the MO-PSO with 50 generations, 50 archive solutions but only 30 particles. The result of this test is illustrated in Figure 22. We remark that there is an accumulation of many solutions in a certain range of QoS parameters. This is due to the fact that the number of particles is not big enough to enable the swarm to fly sufficiently.

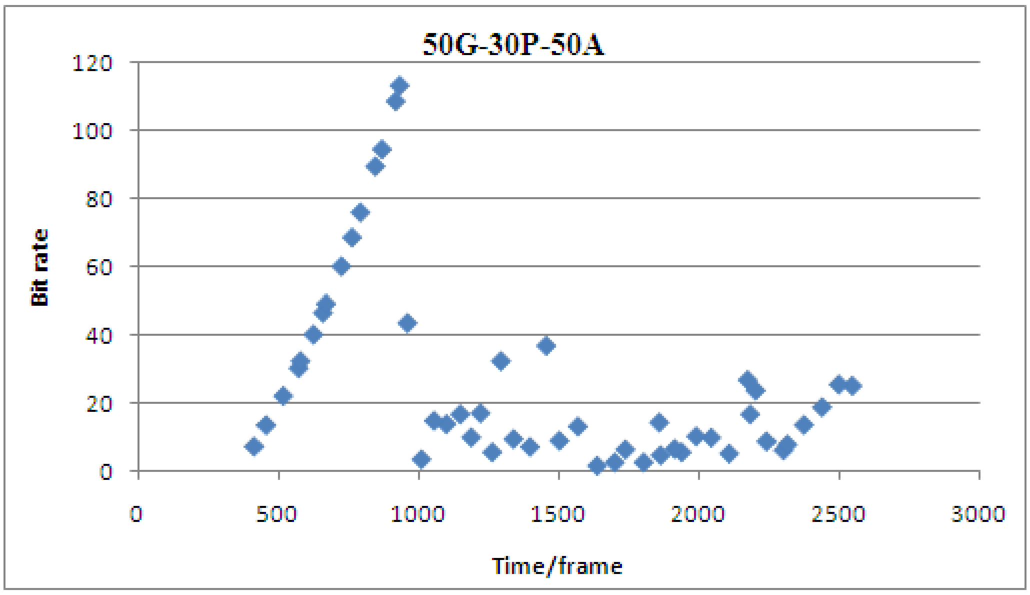


Figure 22. *MO-PSO pareto front for 50G-30P-50A*

Therefore, we perform a second test with a greater number of particles. Figure 23 illustrates the generated pareto front with 50 particles, 50 generations and 50 archive solutions. We obtain a better result, illustrated in Figure 23, where the solutions are more spread out and the pareto front is finer.

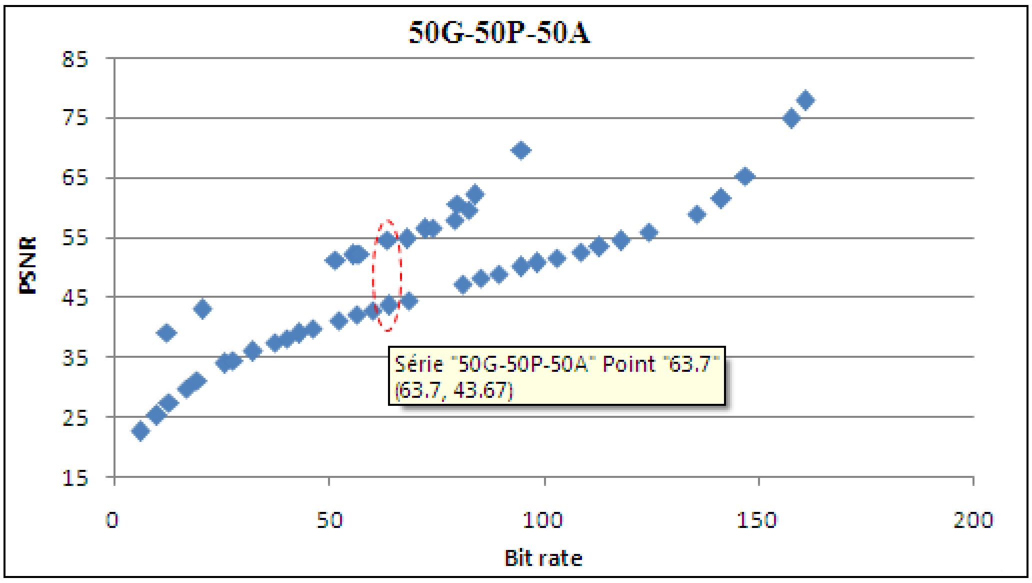


Figure 23. *MO-PSO pareto front for 50G-50P-50A, bit rate vs. PSNR*

However, we remark that sometimes the MO-PSO preserves solutions with different PSNR for nearly the same bit rate (e.g. solutions encircled on Figure 23 have different PSNR for the same bit rate 63.7kb/s). This is explained by the fact that we optimize three objective functions at a time: maximizing PSNR while minimizing bit rate and encoding time. Solutions with lower PSNR are then preserved by the MO-PSO because they offer lower encoding time. This is clearly illustrated in Figure 24 which presents the bit rate in function of the encoding time per frame.

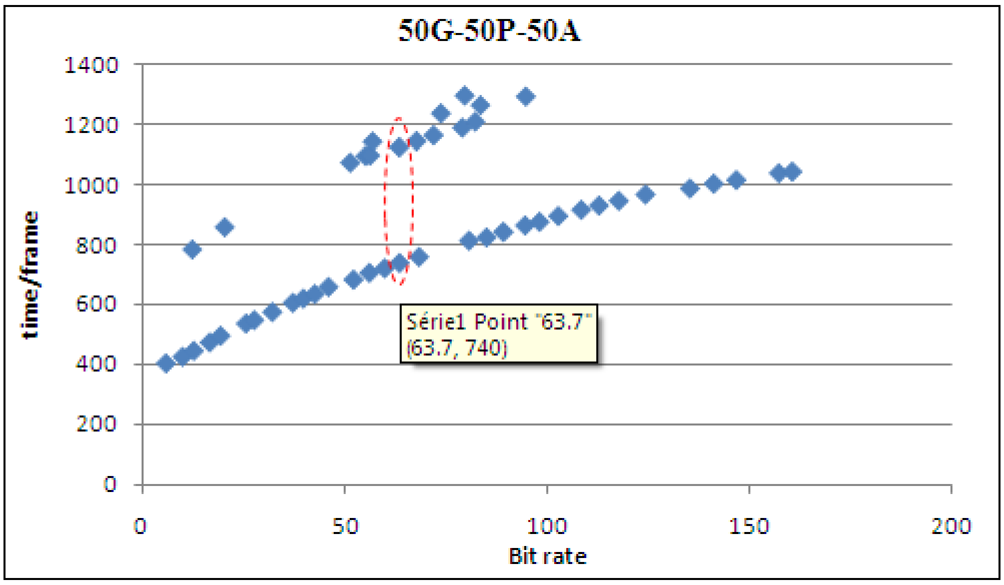


Figure 24. *MO-PSO pareto front for 50G-50P-50A, bit rate vs. encoding time/frame*

This figure shows the coordinates of the same solution depicted in the previous figure which offers lower PSNR but lower encoding time than the solution that is encircled with (i.e. Bit rate=63.7kb/s, PSNR=43.67db and Time=740).

We perform a third test with the same previous number of generations and particles, but higher archive size. A problem that can be encountered with such configuration is that the generated optimal solutions are too close to each others, i.e. QoS parameters are so close that we obtain many configurations covering a limited range of QoS levels and with reduced tolerance intervals. Therefore, in order to enlarge the covered QoS level range and space out the retained solutions, we have to tune the size of the optimal solutions' archive.

To conclude, the results obtained from the second test are enough convenient for our adaptation needs. In fact, it gives us a set of configurations that covers better QoS levels ranges and different scenarios of resources variation: only CPU constraint, only bandwidth constraint, or both CPU and bandwidth constraints.

The generated QoS levels are saved to a file that we call “cfgTable_file.txt”. Each configuration is represented by three fields, which are the maximum number of parameters, the set of configuration parameters and the set of QoS parameters. The first field is used to enable the support of configurations with different configuration parameter sets.

For example, the first QoS level is (2, 1, 50, 15, 1, 0, 1, 22.57, 5.9, 405). This means that it uses the Simplified UMHexagon Search mode, codes every frame in intra-coding, with a QP for I slice equals 50 and without 4x8 inter prediction mode. The corresponding average encoding time per frame is 405 ms, the bit rate is about 6 Kb/s and the PSNR is 22.57 db. The configuration table file is then added to the set of files needed for the functioning of the adaptive encoder.

5. Implementation and test of the online adaptation

5.1 Adaptation API

As we already mentioned in the previous chapter, the adaptation API[4] is composed of two parts: an application independent part which manages resource variation and an application specific part that is hooked into the application to enable its reconfiguration. Table 4 presents the main adaptation functions. The two last functions of the table, setNewConfig() and reconfigure_H264_app() represent the application specific code.

[4] Application Programming Interface

Table 4. *The new adaptation API*

Operation	Description
taskEntry(const char *name, float time, float minT, float period, float desiredBR, float minBR, int adapt)	Manages the task entry event. It adjusts CPU and bandwidth budgets of adaptive tasks, then claims the application reconfiguration if necessary.
taskExit (int taskIndex)	Manages the task exit event. It adjusts both CPU and bandwidth budgets of adaptive tasks, then claims the application reconfiguration if necessary.
BW_controller()	Triggers a bandwidth variation event.
BW_request(float BW_constraint)	Manages the bandwidth variation event. It adjusts the bandwidth budgets of adaptive tasks, then claims the application reconfiguration if necessary.
setNewConfig(EncoderParams *p_Enc)	Searches for the best configuration and triggers reconfiguration.
reconfigure_H264_app(EncoderParams *p_Enc)	Reconfigures the H264/AVC encoder with the new configuration.

In order to enable dynamic reconfiguration of the JM encoder, we select the best configuration that respects the incoming constraint, and then we save it to a file "reconfigure.cfg" to be read as a new configuration file [47].

5.2 Example of adaptation test

We present hereafter a test scenario and the results of the adaptation technique. We focus on the resources allocation evolution and the reconfiguration behavior following the resources variation events.

5.2.1 *Test scenario*

Let’s consider a scenario of system task set with different resource variations (task entry, bandwidth variation, and task exit). Table 5 presents the list of tasks that enters (event TE) or exits (event TX) from the system with the corresponding time of occurrence (the frame number). We consider a set of 4 tasks: a video encoder task, H264_enc, and 3 other multimedia tasks MM2, MM3, and MM4. The MM2 task is non-adaptive and the three others are adaptive ones.

Table 5. *A task set scenario*

Nb frame	Ev	Ta	adaptive	Texe(ms)	MinTexe (ms)	Period (ms)	BR(kb/s)	MinBR(kb/s)
0	T	H2	1	500	100	500	1400	28
32	T	M	1	300	100	300	800	300
60	TE	M	0	30	30	100	50	50
120	TE	M	1	100	80	300	500	100
140	T	M	-	-	-	-	-	-
220	T	M	-	-	-	-	-	-

5.2.2 *Validation of the resource allocation*

We depict in Table 6 the evolution of the H264 task resource budgets after each resource variation event. We give the amounts of allocated bandwidth and CPU time. We also give the total adaptive CPU and bandwidth budgets evolution. The last line of the table presents the selected configuration number in order to illustrate the reconfiguration occurrence.

The first task that enters the system is the H264 encoder. It is allocated the totality of the resources (500ms, 1400kb/s). The entry of the second task MM2, which occurs at frame 32 encoding time, causes a substantial decrease in the CPU and bandwidth budgets of the H264 task. They drop to about the half. This is because the new task MM2 requires high resource allocation (required CPUbgt =1, required BWbgt = 0.571). As a consequence, a reconfiguration is needed to decrease the quality level of the encoder. Later in Figure 28, we can notice a difference between the video quality before (image (a)) and after (image (b)) MM2 entry.

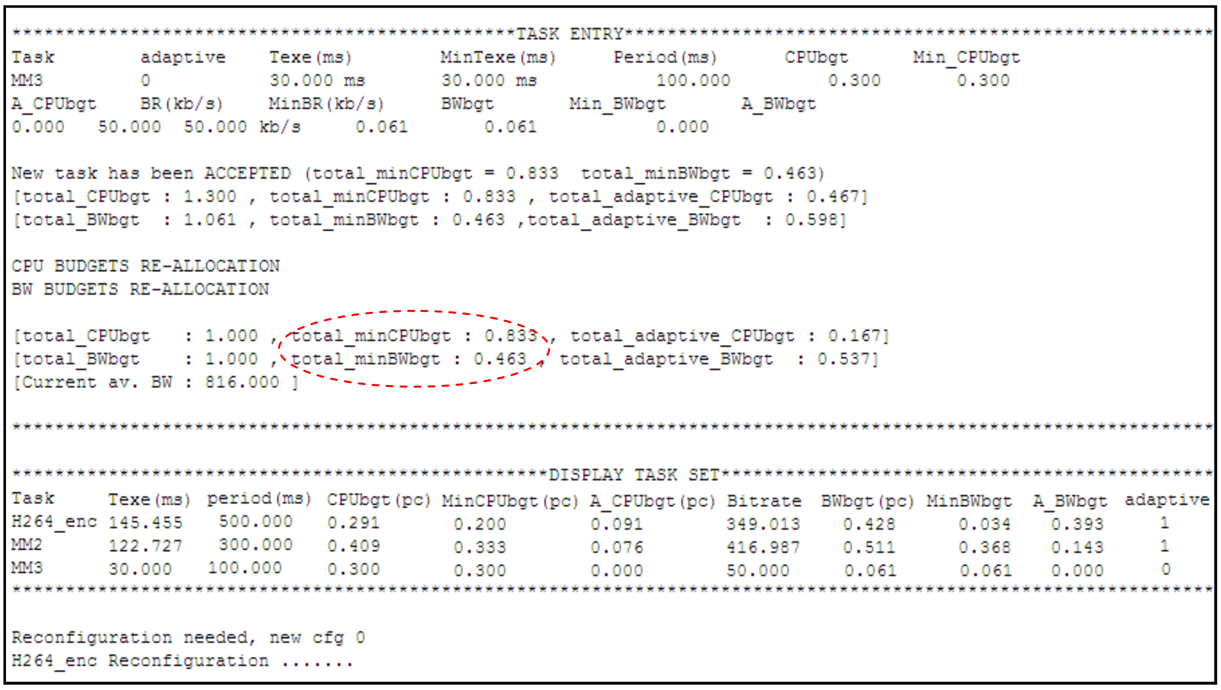
Table 6. *The H264 resources allocation and reconfiguration*

Events	**H**	**M**	**B**	**M**	**B**	**B**	**M**	**T**	**B**	**B**	**B**	**T**	**B**
Nb Frame	0	3	4	6	8	1	1	1	1	1	1	2	2
Tot. BW	**1**	**1**	**8**	**8**	**7**	**8**	**8**	**8**	**1**	**9**	**1**	**1**	**9**
Texe	5	2	2	1	1	1	1	3	3	3	3	5	5
CPU bgt	1 ↘	0	0 ↘	0	0	0	0 ↗	0	0	0	0 ↗	1	1
Bitrate	1	8	3	3	3	3	3	8	9	9	1	1	9
BWbgt	1 ↘	0	0 ↘	0 ↘	0 ↗	0	0 ↗	0 ↗	0 ↘	0	0 ↗	1	1

Tot adaptive CPU bgt	0	0	0	0	0	0	0	0	0	0	0	0	0
Tot adaptive BW bgt	0	0	0	0	0	0	0	0	0	0	0	0	0
Config.	**9**	**3**	**1**	**0**	**0**	**0**	**0**	**5**	**7**	**7**	**8**	**8**	**7**

When the third task MM3 enters, the CPU budget of the H264 task drops to 0.291 (i.e. 145.455ms), and its bandwidth budget becomes 0.428 (i.e. 349.013Kb/s). The worst configuration (cfg.0) existing in the configuration table we use offers a minimum encoding time of about 200ms with a bit rate of nearly 222Kb/s. In this case, even if cfg.0 gives an encoding time higher than the requested one, we accept it as a new configuration for the new set of the allocated resources budgets. This configuration selection remains until the first task exit occurs at frame 140 encoding time. At that moment, the CPU and bandwidth budgets allocated to the encoder increase greatly (70% of CPU time and 94.2% of bandwidth). Figure

25 depicts the resource allocation and the different total budgets values for the task set that is obtained just after the entry of the task MM3.

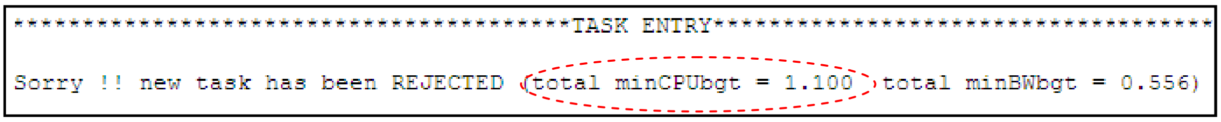


```
*********************************************TASK ENTRY*********************************************************
Task        adaptive    Texe(ms)        MinTexe(ms)      Period(ms)       CPUbgt      Min_CPUbgt
MM3         0           30.000 ms       30.000 ms           100.000          0.300        0.300
A_CPUbgt    BR(kb/s)    MinBR(kb/s)     BWbgt       Min_BWbgt       A_BWbgt
0.000    50.000   50.000 kb/s     0.061       0.061           0.000

New task has been ACCEPTED (total_minCPUbgt = 0.833  total_minBWbgt = 0.463)
[total_CPUbgt : 1.300 , total_minCPUbgt : 0.833 , total_adaptive_CPUbgt : 0.467]
[total_BWbgt  : 1.061 , total_minBWbgt : 0.463 ,total_adaptive_BWbgt  : 0.598]

CPU BUDGETS RE-ALLOCATION
BW BUDGETS RE-ALLOCATION

[total_CPUbgt    : 1.000 , total_minCPUbgt : 0.833 , total_adaptive_CPUbgt : 0.167]
[total_BWbgt     : 1.000 , total_minBWbgt : 0.463 , total_adaptive_BWbgt  : 0.537]
[Current av. BW : 816.000 ]

****************************************************************************************************************

**************************************************DISPLAY TASK SET**********************************************
Task     Texe(ms)  period(ms)  CPUbgt(pc) MinCPUbgt(pc) A_CPUbgt(pc) Bitrate  BWbgt(pc) MinBWbgt  A_BWbgt  adaptive
H264_enc 145.455    500.000    0.291        0.200         0.091       349.013    0.428     0.034     0.393     1
MM2      122.727    300.000    0.409        0.333         0.076       416.987    0.511     0.368     0.143     1
MM3      30.000    100.000     0.300        0.300         0.000       50.000     0.061     0.061     0.000     0
****************************************************************************************************************

Reconfiguration needed, new cfg 0
H264_enc Reconfiguration .......
```

Figure 25. *Example of the system task set status (after 3 task entries)*

We notice that the non-adaptive task MM3 is allocated the entire requested resource budgets on top of the figure. We also notice that the sum of each resource allocated budgets is equal to 1, i.e. the resources are fully utilized with respect of tasks timing constraints. We remark as well that the total minimum budgets are less than 1. That means that the system is able to host more tasks but with lower quality of service.

However, when the user desires to enter the fourth task MM4 the system reject this request. This is because the CPU load will be so important that the processor will not be able to guarantee the minimum requested resources for all tasks. This is indicated in Figure 26 which illustrates that the calculated total minimum CPU budget with task MM4 exceeds 1.



```
*****************************************TASK ENTRY*************************************

Sorry !! new task has been REJECTED (total minCPUbgt = 1.100  total minBWbgt = 0.556)
```

Figure 26. *New task denial*

As for the total adaptive budgets, it is obvious from the table that they are proportional to the available resources. Whenever there is a resource scarce, the adaptive tasks have their budgets decrease and therefore their ability of adaptation decreases too.

5.2.3 Reconfiguration behavior

We summarize in the curves depicted by Figure 27 the bandwidth variation during the test scenario. We indicate with colored points the occurrence of the H264 encoder reconfiguration.

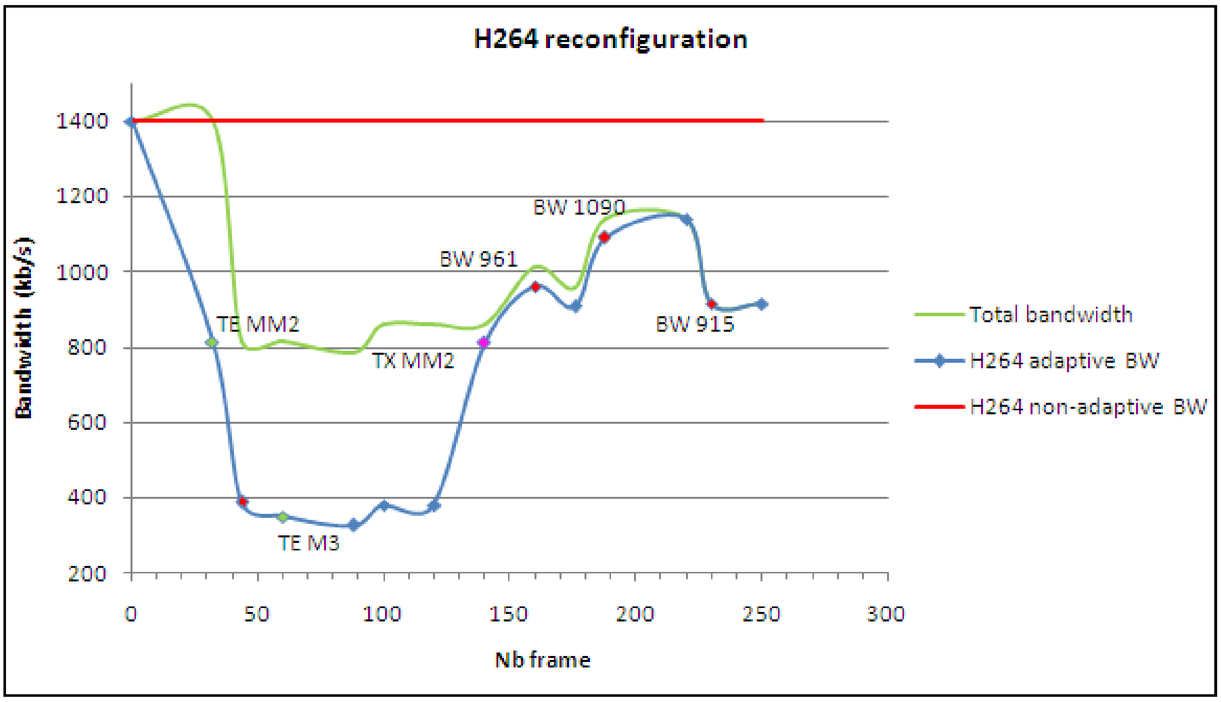


Figure 27. *Reconfiguration behavior of the encoder following resource variations*

We designate by the green point a reconfiguration due to a task entry event. In our case, the encoder needs to reconfigure in both MM2 and MM3 task entries because these need important CPU and bandwidth resources. The reconfigurations due to bandwidth variation are designated by the red points. We remark that the application doesn't reconfigure at every bandwidth variation and thus is able to achieve a stable status during different intervals not only when the environment is stable but also when there are small resource variations. This is thanks to the interval tolerance that has been implemented in the application adaptor.

The red line designates the requested bandwidth value of a non adaptive video encoder during the test scenario. It is obvious that without the adaptation mechanism the application bit rate usually exceeds enormously the available bandwidth. The output video will consequently present many problems such as jitter, network delay, and packet loss.

5.2.4 Impact of the adaptation on video quality

Every resource variation event that provokes a reconfiguration of the H264 encoder induces a change in the resulted video quality. We illustrate in Figure 28 the major quality variation of the encoded video. We use the test video “foreman.yuv” with a QCIF format.

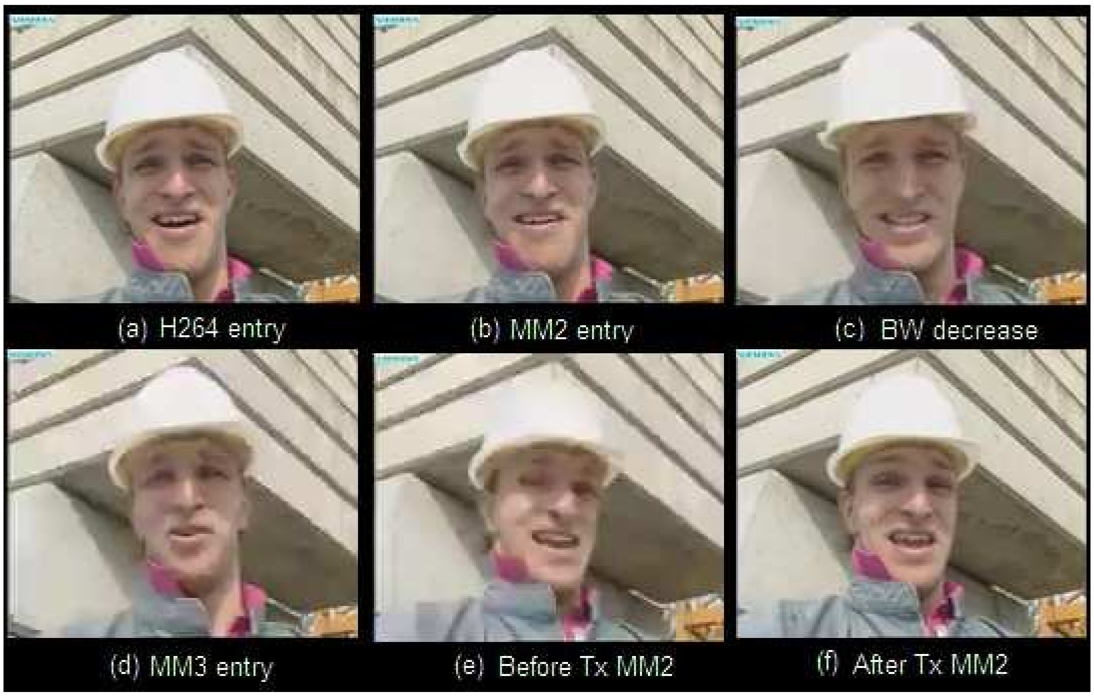


Figure 28. *Impact of adaptation on video quality*

We notice that the encoder manages to scale 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)). However, it is clearly depicted in the image (f) that a decision of reconfiguration has been taken to upgrade the video quality when an increase of the system available resources occurs (task MM2 exits).

5.3 Adaptation overhead

We calculate the average overheads of the different adaptation modules using a HP laptop with a 1.78GHz processor. They are depicted in Table 7. These overheads are negligible compared to CPU intensive multimedia applications (up to 500ms in our case) which, indeed, tolerate constraints exceeding to some extent.

Table 7. *Adaptation overhead*

Adaptation module	Overhead (ms)
Adjust Bandwidth budgets	0.07
Adjust CPU budgets	0.07
Reconfigure H264 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.

6. Conclusion

In this chapter, we presented the implementation steps of our adaptation technique using the H264/AVC video coding as a multimedia CPU-bandwidth intensive case study application.

As a first part, we defined the application QoS parameters and determined the set of configuration parameters to be mainly considered during QoS levels generation. Then, we detailed the MO-PSO/H264 interfacing in order to generate the QoS levels table.

In the second part, we presented the implementation and test results of the adaptation mechanism. We proposed a test scenario composed of different resource variation events and presented the effect of adaptation on resource allocation and configuration behavior of the H264 task.

The results confirm that the prototype is able to respond to resource constraints while maintaining a minimum of acceptable QoS and a continuous service with stability intervals. Such results uphold the importance of the application-aware adaptation.

Conclusion & perspectives

Embedded systems are often subject to resource constraints such as limited battery life, narrow memory space, and environment variations such as sudden and unpredictable network load. Thus, they are asked to adapt internally to their limited computational and energy resources as well as to changes in the external environment in order to support multimedia quality of service (QoS).

Many works in the literature have dealt with adaptation. Researchers have proposed adaptations in several system layers (hardware, application, operating system, and network), different time granularities (coarse and fine-time granularities), and under various sets of constraints (energy, CPU time, real-time constraint, network bandwidth). A widely used adaptation technique to save energy and preserve performance is to dynamically reconfigure the system resources in response to changes in application demands and resource availability.

It is important to note that the adaptation work does not attempt to provide resource guarantees to applications. Such guarantees, typically encountered in real-time systems, require guarantees from lower layers of the system. But the environment of a mobile computer is too unpredictable for such guarantees. Hence, such work only promises to inform applications when their environment changes, arbitrate between applications competing for scarce resources, and reconfigure the application to meet resource constraints.

In this work, we have proposed an adaptive system structure that allows the system to adapt to i) external changes due to its external environment; we considered the available network bandwidth, and ii) CPU load that varies with the number of running applications; it coordinates the adaptation of CPU time depending on the number of running applications. These changes adjust the settings of applications in response to the constraints of resources while maximizing the QoS.
We applied those changes on a complex multimedia application, the video compression with the H264/AVC standard. The adaptation technique was tested using an automatically generated configuration database for the H264/AVC standard.

Tests are made by a modification of the system load and the network bandwidth. The evaluation confirms that the technique does a good job of balancing adaptation to resource constraints and stability of the system. It shows that thanks to the adaptation mechanism, the system is able to scale its output quality according to the environment fluctuation while guaranteeing the continuity of the service. It also upholds the importance of applying adaptations in the application level.

At the same time, our model suggests avenues for further improvements. Concerning the adaptation aspect, we plan as next steps to map the application on an RTOS and run it on the target system to test its auto-adaptation and concurrency management capabilities. A more thorough study of the system stability is needed. As for the target multimedia application, we consider looking more in depth for better metric combining different QoS parameters to better quantify the video quality.

References

[1] S. Mohapatra and N. Venkatasubtramanian, *"Power-Aware Reconfigure Middleware,"* Proc. 23rd IEEE Int'l Conf. Distributed Computing Systems, May 2003

[2] Pillai, P., and Shin, K. G. *"Real-time dynamic voltage scaling for low-power embedded operating systems"*. In. Proc. of the 18th ACM Symp. on Operating Systems Principles, 2001

[3] Padmanabhan Pillai and Kang G. Shin, *"Real-time dynamic voltage scaling for low-power embedded operating systems,"* Proceedings of the eighteenth ACM symposium on Operating systems principles

[4] 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

[5] A. Vahdat, A. Lebeck, and C. Ellis, *"Every joule is precious: A case for revisiting operating system design for energy efficiency,"* in Proc. of 9th ACM SIGOPS European Workshop, Kolding, Denmark, Sept. 2000.

[6] Wanghong Yuan, Klara Nahrstedt, Sarita V. Adve, Douglas L. Jones, and Robin H. Kravets, *"GRACE-1: Cross-Layer Adaptation for Multimedia Quality and Battery Energy,"* IEEE transactions on mobile computing, Vol. 5, No. 7, Jul. 2006

[7] H. H. Chu and K. Nahrstedt, *"CPU service classes for multimedia applications,"* in Proc. of IEEE Int. Conf. On Multimedia Computing and Systems (ICMCS'99), Florence, Italy, pp. 296–301, Jun. 1999

[8] S. Banachowski and S. Brandt, *"The best scheduler for integrated processing of best-effort and soft real-time processes,"* in Proc. of SPIE Multimedia Computing and Networking Conference, San Jose, CA, Jan. 2002

[9] W. Yuan, and K. Nahrstedt, *"Energy-Efficient CPU Scheduling for Multimedia Applications,"* ACM Transactions on Computer Systems, Vol. 24, No. 3, Pages 292–331, Aug. 2006

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

[11] M. Mesarina and Y. Turner, *"Reduced energy decoding of MPEG streams,"* in Proc. of SPIE Multimedia Computing and Networking Conference, San Jose, CA, Jan. 2002

[12] N. Pham Ngoc, W. van Raemdonck, G. Lafruit, G. Deconinck, and R. Lauwereins, *"A QoS framework for interactive 3D applications,"* Proceedings of the ninth international conference on 3D Web technology, 2004

[13] Shih-Chang Hsia, *"An adaptive video coding control scheme for Real-time MPEG applications,"* EURASIP Journal on Applied Signal Processing, 2003

[14] Peng Shaomin, Van Zon Cornelis C., Zhong, Zhun, *"Resource scalable decoding,"* US Patent 6704362, Mar. 9, 2004

[15] Damir Isovié and Gerhard Fohler, "*Quality aware MPEG-2 Stream Adaptation in Resource Constrained Systems,*" Proceedings of the 12th 16th Euromicro Conference on Real-Time Systems (ECRTS'04), 2004

[16] S. Saponara and L. Fanucci, *"Data-adaptive motion estimation algorithm and VLSI architecture design for low-power video systems,"* IEEE Proc.-Comput. Digit. Tech., Vol. 151, No. 1, Jan. 2004

[17] 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

[18] P.I. Hosur and K.K. Ma, *"Motion Vector Field Adaptive Fast Motion Estimation,"* Second International Conference on Information, Communications and Signal Processing (ICICS '99), Singapore, Dec. 7-10, 1999

[19] Alexis M. Tourapis, Oscar C. Au, Ming L. Liou, *"Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation,"* Department of Electrical and Electronic Engineering, The Hong Kong University of Science and Technology

[20] W. Il Choi, J. Lee, S. Yang, and B. Jeon, *"Fast motion estimation and mode decision with variable motion block sizes,"* Proceedings of SPIE, Vol. 5150, 2003

[21] Jiancong Luo, Ishfaq Ahmad, Yu Sun and Yongfang Liang, "*A multistage fast motion estimation scheme for video compression,*" International Conference on Image Processing (ICIP), 2004

[22] A. K. Kannur, Baoxin Li, "*Power-aware content-adaptive H.264 video encoding,*" IEEE International Conference on Acoustics, Speech and Signal Processing, pp.925-928, 2009

[23] M. Satyanarayanan, Brian Noble, Puneet Kumar, and Morgan Price, *“Application-Aware Adaptation for Mobile Computing,”* ACM SIGOPS Operating Systems Review , Volume 29 Issue 1, Jan. 1995

[24] S. Mohapatra and N. Venkatasubtramanian, *“Power-Aware Reconfigure Middleware,”* Proc. 23rd IEEE Int’l Conf. Distributed Computing Systems, May 2003

[25] GRACE-2: 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

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

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

[28] K. Loukil, N. Ben Amor, **M. Ben Saïd**, M. Abid, *“OS service update for an online adaptive embedded multimedia system,”* IEEE Symposium on Computers and Communications (ISCC'09), Sousse, Tunisia, Jul. 5-8, 2009

[29] **M. Ben Saïd**, K. Loukil, N. Ben Amor, Jean Philippe Diguet, M. Abid, *“A timing constraints control technique for embedded real time systems,”* International conference on Design & Technology of Integrated Systems in nanoscale era (DTIS'10), Hammamet, Tunisia, Mar. 23-25, 2010

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

[31] 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.

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

[33] 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

[34] J. Kennedy and R. Eberhart, *"Particle swarm optimization."* Proc. IEEE International Conf. on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, 1995

[35] Russell Eberhart and James Kennedy, *"A New Optimizer Using Particle Swarm Theory,"* Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995

[36] 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

[37] Baochun Li and Klara Nahrstedt , *"A Control-Based Middleware Framework for Quality of Service Adaptations,"* IEEE journal on selected areas in communication, Vol. 17, No. 9, 1632-1650, Sept. 1999

[38] Cristian Koliver, Jean-Marie Farines, and Klara Nahrstedt, *"QoS Adaptation Based on Fuzzy Theory,"* 2002

[39] Jörn Ostermann, Jan Bormans, Peter List, Detlev Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer, and Thomas Wedi, *"Video coding with H.264/AVC: Tools, Performance, and Complexity,"* IEEE circuits and systems magazine, 2004

[40] Thomas Wiegand, Heiko Schwarz, Anthony Joch, Faouzi Kossentini, and Gary J. Sullivan, *"Rate-Constrained Coder Control and Comparison of Video Coding Standards,"* IEEE transactions on circuits and systems for video technology, Jul. 2003

[41] Jeremiah Golston, Distinguished Member, Technical Staff, Texas Instruments, Dr. Ajit Rao Member Group Technical Staff, Texas Instruments, *"Video Compression: System Trade-Offs with H.264, VC-1 and Other Advanced CODECs"*, White paper, Texas Instruments, Aug. 2006

[42] Telecommunication standardization sector of ITU, "Advanced video coding for generic audiovisual services", Mar. 2005

[43] Feng Pan, *"Digital Video Coding – Techniques and Standards,"* Studies in Computational Intelligence (SCI) 58, 13–53, Springer-Verlag Berlin Heidelberg, 2007

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

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

[46] 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

[47] Fatma Ben Taher, *"implementation of adaptation techniques on the H264/AVC application,"* End of study memoire in computer science engineering, ENIS, 2010