Abstract

Applications of high-quality multimedia embedded systems (HQMES) contain concurrent functionality pieces; they may have a flexible structure consisting of services, components, or tasks. This allows applications to offer different execution possibilities to provide a same (or different) output quality. Since multimedia applications are typically greedy resource consumers, they may interfere another’s execution; this can be a source of unpredictability, especially in the event of reconfigurations. This paper presents a description on how to minimize such execution uncertainty at different levels: (1) the OS has to offer the basic real-time support, (2) appropriate resource management techniques will have to be built, usually inside a QoS RM (QoS resource manager entity), and (3) applications should be designed in a flexible but yet manageable way. This paper recovers an efficient application characterization as the basis for offering a manageable application structure. On top of this application characterization, a contract model is described to provide predictable execution. Also, two reconfiguration algorithms are presented, so that application execution is kept predictable even during reconfiguration.

1. Introduction

High-quality multimedia systems are user-oriented applications that often have strong needs for reconfiguration of their functionality and/or execution. On one hand, functional flexibility allows applications to offer a range of output delivery possibilities to the final user. On another hand, operational flexibility relates to the appropriate design of the application structure supported by the platform in a way that different execution modes are possible to deliver different output qualities. Therefore, it is necessary that applications and the platform are designed in a way that reconfiguration situations are possible within given time bounds that do not annoy the user perception. HQMES have also high requirements for real-time operation as, for example, high-quality video rendering. These systems require efficient support from the underlying platform. Not only the network protocols must be deterministic, but also the operating system and distribution middleware. In this context, it is required the mixing of deterministic low-level mechanisms with efficient high-level protocols and strategies. One without the other may result in useless isolated effort [3,4]. Lower-level mechanisms will provide the basics to build application execution isolation. Low-level protocols are scheduling of the resources, mainly processor, memory, communication media, and currently also battery. Above the real-time operating system level, there are the policies and mechanisms to make these environments more flexible; examples of these are admission control based on application compatibility and resource availability, mode change protocols as the basics for application and system reconfiguration, and monitoring of application behavior regarding resource consumption. Application and system reconfiguration in HQMES cannot tolerate the introduction of operation uncertainty. The system should be kept under predictable operation conditions even in the event of reconfiguration. As a consequence, real-time techniques and predictable resource management will have to be used.

\footnote{This work has been partially funded by ARTISTDesign NoE (IST-2007-214373) and the Spanish Ministry of Tourism, Commerce, and Industry (TSI-020501-2008-159).}
This paper puts into context the traditional problems of these systems and introduces how a contract scheme can guarantee resource assignments to applications to preserve time predictability even during reconfiguration. Therefore, this paper presents the basics of what contract-based execution has to offer, and it also shows two mode change algorithms as the basis for application and system reconfiguration.

The paper is structured as follows. Section 2 offers an overview of related work. Section 3 presents the basic principles for predictable resource management. Section 4 describes an approach for complete application modeling that offers the QoSRM a complete system level view to manage application execution. Section 5 presents the reconfiguration algorithms that are designed for offering efficient transitions among application states whenever they should change to execute a different service/component/task set. In section 6, the validation of these ideas is presented. Section 7 presents the conclusions of the work.

2. Background on Predictable Resource Management in Multimedia Applications

Best effort techniques have appeared over the years for general purpose distributed multimedia domain based on implementing coordination entities that cooperate with multimedia applications as opposed to enforcing any behavior on them. Also, other approaches related to flexible scheduling techniques have appeared; however, these lack sufficient abstraction level to cover all needs of multimedia applications, not offering application models nor including enough application semantics. The classical scheduling techniques usually provide temporal isolation; however, it is required to raise the reasoning level to present QoS manager entities that take into account a small degree of semantics of multimedia applications. From a more integral perspective, approaches as [4] offered layered architectural views for implementing manager entities in HQMES by means of defining the appropriate hooks to include deterministic resource management mechanisms.

Moreover, reconfiguration needs of these systems have mostly been left aside. Nowadays, reconfiguration concerns are appearing related to the networked embedded systems, mostly in the context of cyberphysical systems. Only few research addressed mode changes mainly for hard real-time systems [5]; these are performed at task level, including few application semantics.

Therefore, it has not been addressed how to minimize unpredictability in HQMES, even during reconfiguration. This paper targets this problem by means of applying a contract model, refreshing an application characterization model [4] that allows the manager entity (QoSRM) to efficiently arbitrate application execution. Moreover, it presents two algorithms for performing basic system level reconfiguration. For example, this may be the case of an application that is requested by the user to change from a small image size to a full screen size, which will require a different higher processing power and resources from the system.

3. Principles of execution of QoS-based resource managers

The following operations must be supported by a QoSRM:

- **Admission control.** Application admission will be checked against resource availability.
- **Resource reservation guarantees.** Applications are given strict guarantees on the availability of the resource budgets contracted with the system.
- **Scheduling and management of resources.** To guarantee sufficient resources, the system must arbitrate application execution.
- **Resource usage monitoring.** It allows performing accounting of the usage that applications make of the system resources; it allows detecting situations of over or infra utilization of resources as a first step to undertake corrective actions.
- **Dynamic adaptation.** At run time, the system must be able to adapt to changes as, for instance, instant variations of computing demands of the incoming media.
- **Stability.** The system must be able to react in a stable way to any external or internal event; reaction to them should not be a source of instability.
- **Efficient transitions.** Mode changes of applications (such as a switch of quality level) may be frequent, depending on, for instance, the user action. Such changes determine a new configuration of the system that involves a new distribution of resources among applications.
- **Global optimization.** Appropriate resource assignment must be carried out in a way that the overall quality offered by applications and the overall quality perceived by the user is maximized.
- **Handling overload, faults, and alarms.** The system must be able to filter out overload situations (which are very frequent in the multimedia processing
environment), to recover from faults, and to handle alarms that may be triggered during system operation.

- Estimation of resource requirements of applications. Resource requirements for media processing algorithms on HQMES are estimated by application experts, since resource demands are variable and data dependent.

Currently, there is no integrated solution that is capable of fulfilling all these principles, in fact, only a few solutions address satisfactorily three (four, at most) of them in a integrated way.

**Architectural principles for QoSRM implementation**

The presented approach follows the architectural principles of HOLA-QoS [4], shown in figure 1. This architecture, centralized in its initial version, has also been ported to a multiprocessor architecture, and it is currently being ported to a fully open distributed environment with real-time virtual machine structure.

![Figure 1: HOLA-QoS Architecture](image)

Some of the architecture characteristics are:

- It has a hierarchical multi-level structure that allows working at **different abstraction levels**. Integrating a different reconfiguration protocol requires only to code it in the appropriate layer, precisely, layer 2.

- Its homogeneous layer structure allows for easy replacement of components in the layers to experiment with different resource management mechanisms and reconfiguration policies, which is an additional detail related to the previous characteristic.

- Its control is hierarchical: (1) upper layers perform less frequent control operations although such operations are the ones that have most influence on the system as a whole, whereas (2) lower layers perform the more automatic control mechanisms (such as resource accounting and monitoring, resource enforcement, and admission control).

**4. Application Characterization Approach**

This section presents a refreshed approach for modeling applications in HQMES. The basic principles for it are the following:

- Adjusting to the natural structure of multimedia applications: (1) filter-based media processing, and (2) model the different delivery output qualities (quality levels) that a multimedia application may produce as results (i.e., different image resolutions, different image sizes, etc.).

- Integrating smoothly the hooks for performing system-wide and cross-application resource management.

High quality video streaming applications have a pipeline structure. In them, the output quality is directly influenced by the number and complexity of the intermediate filters to be applied to the data. Since application tasks are greedy [2], output quality is related to the amount of resources that multimedia filters consume.

In our approach, applications have two main parts with separate responsibilities: the control part for interaction with the QoSRM, and the functional part that executes the media processing algorithms.

The structure of the functional part is shown in figure 2 for a typical high-quality video processing application. Applications receive data (input signal), process it, and produce an output signal for the final user. Such processing is carried out in a continuous way, i.e., as long as tasks have input data to process, they will apply filters or algorithms to the received data in a non-stop fashion.

![Figure 2. Internal structure of an application](image)

The control part is an additional task in charge of receiving the control commands from the QoSRM and restructuring the functional part as necessary. In the event of a notification from the QoSRM to an application for it to switch to a different quality level, the control part of the application will have to take appropriate action. Such a quality level change may require that the current task set is changed or parameterized in a different way since the new quality level may have different resource requirements. The application control part will have to parameterise the code of the required tasks and it will change the structure of the application functional part, i.e., the connections between tasks and/or filters, if necessary.
Figure 3 presents the application model. The QoSRM has the whole system information in its internal repository (in the case of HOLA-QoS, this is the system profile). The application model follows these criteria:

- Each application can deliver different output results or quality levels.
- A quality level may be implemented with different resource combinations, each called QLConfiguration (i.e., to deliver a full screen size might be possible for an application with $c_i$ processor percentage and $y_i$ memory or compensating memory with processor as $c_i + \nu$ processor and $y_i - \nu$ memory).
- Each realization of a quality level can be achieved by a different set of application tasks (Task).
- Tasks may be grouped (Cluster) in a way that they compensate one another in the usage of resources.
- Each cluster, which is a group of tasks with similar properties for purposes of compensation, can be realized by assigning a different set of resource budgets (Budget) to it for each resource it uses. This is represented as a ClusterConfiguration.
- Also, a certain tasks can take different versions or profiles (TaskConfiguration) depending on the input data it has to process.

5. Contract-Based Execution

A contract is a firm agreement involving the QoSRM and the applications so that: (1) the QoSRM guarantees applications the computation resources that they require in order to deliver a given output quality level, and (2) applications must provide a certain output quality with the agreed resource amounts. Under a contract model, responsibility is split so that application programmers design the applications in such a way that quality of the outputs are guaranteed if enough computation resources are available. Application experts have complete knowledge for this; it is especially true for the case of high quality video streaming applications.

The basic technique to implement the contract model is the budget enforcement mechanism. Assigning resource budgets to applications and enforcing them avoids interference of the greedy multimedia applications, providing application execution isolation. Guaranteeing budgets to application tasks is done by means of forcing tasks not to use more resources than they have contracted. Therefore, even if greedy tasks are willing to perform budget overruns, the QoSRM will steal the processor from them whenever they have consumed the budget.

Guaranteeing resource budgets is performed applying: (1) resource usage monitoring and accounting, and (2) application prioritization. The QoSRM manages application execution according to task priorities (which come from the importance of their application). Resource usage accounting is a constant procedure done by the QoS RM in its lower level mapped on the RTOS primitives to access consumed processor time. Every task execution is checked against overruns when entering and when leaving the processor.

HOLA-QoS defines the admission component, monitoring component and the configuration setting component that define the interfaces that should be implemented to implement the contract model. The admission component defines the interfaces for requesting admission and negotiating assignments of resource budgets to applications, clusters, and tasks. Monitoring component defines the interfaces for actively performing accounting of resource usage of all tasks (and, therefore, clusters and applications) to detect and prevent budget overruns. The configuration setting component provides the interfaces for enforcing the contracted resource budgets.

6. Reconfiguration scheme

We refer to the current system configuration as being the complete set of task profiles (set of TaskConfiguration objects) that are active in the system. By the user action or by any internal decision of the adaptation policies of the QoSRM, the current system configuration may have to be replaced by a new one. This occurs, for instance, when: (1) the quality level of one application is changed, (2) a new application is launched, (3) an application is stopped,
and (4) changes of input media requiring more computation resources to be assigned to some application.

The replacement of the current configuration of the system requires a protocol to perform a smooth transition between configurations; it must be able to manage the transition not annoying the user.

Termination of tasks is also another factor to consider. In mode change algorithms for hard real-time systems, it is usual practice that tasks that will not be part of the new mode are allowed to terminate normally. This simplifies their programming. For HQMES, this work has set two termination criteria for tasks of the old mode. Such criteria depend on the application type and its degree of tolerance to data loss in the transition from the old mode to the new one.

Whereas hard real-time systems focus mode change protocols at task level, this scheme falls short for HQMES since it has no information of any higher level application semantics or structure. Following, a set of basic considerations has been identified: (1) applications have to be able to configure at functional level to change their quality level, and (2) the QoSRM shares with each application knowledge of the internal task structure and its resource consumption for each of their possible implementations of quality levels.

The generalized mode change protocol, shown in figure 4, splits the responsibility for the transition between applications and the QoSRM. It assumes that only applications know the precise reconfiguration needs of their internal structure (management of buffer contents and new connections, etc.). Even when diverse multimedia applications have been identified, full characterization of them has still not been achieved; this is the case for 3D graphics applications where any user interaction may require launching a new task to perform some graphics calculation at any time. In this context, the safe approach consists of the separation of responsibilities according to the knowledge that applications and QoSRM have of the system situation and their own functionality.

The QoSRM sends the command to an application to change the current implementation of a quality level. Upon reception of this command, the application is responsible of changing the quality level by its own means; typically, it does it by appropriate parameterization of the code of current tasks or changing the set of tasks that must execute in the new mode and their connections in the data flow. The application reconfigures itself in two steps: (1) stopping the old mode tasks that will not be part of the new mode, and (2) creating the new mode tasks, establishing the appropriate connections for them in the data flow. Once the application has reconfigured to the new quality level implementation, it informs the QoSReM of the result of the reconfiguration. Upon reception of this result information, the QoSReM performs the appropriate operations to set the quality level configuration that, in the end, will require the assignment of the resource budgets for tasks.

The order in which the QoSReM sends the reconfiguration commands to applications may be: (1) by load introduced in the new configuration or quality level (so that applications that lower their quality in the new mode will make the transition first), or (2) by application importance.

An improved version for immediate transitions, for example, in a situation where the input media changes (i.e., the channel is changed) is presented in figure 5.

The parameters and functions presented in the algorithm of figure 5 are:
- \( \Omega \): application set changing to a new system configuration.
- \( OM_a \): task set that implements the old quality level of application \( a \).
- \( NM_a \): task set that implements the new quality level of application \( a \).
- \( Cont_a \): set of tasks that remain in the new quality level; their profile may have to change.
- \( send\_cmd\_reconf(ql\_new) \): the QoSReM sends the reconfiguration trigger (command) to the application.
- \( wait\_complete\_reconf\_ack \): waits until the application has finished its complete functional reconfiguration.
7. Validation

Validation experiments have been carried out that show the feasibility of the execution of a multimedia system managed by a QoSRM that arbitrates the execution of applications with the presented application model. Also, in these experiments mode change measurement results have been obtained to check their suitability. The architecture harness was implemented originally on TriMedia platforms (TM1000 and TM1100) from former Philips Semiconductors, a multi-core platform optimized for high-quality video systems. Currently, the harness architecture and the above mentioned mechanisms have been implemented in Java (RTSJ) and tested on an ARM processor running Red Hat Linux and its real-time patch for TimeSys JVM. For these experiments the full characterization of applications has been utilized. Resource budgets assigned by the system coincide in this case with the required computation time. Results show that budget enforcement mechanisms and mode change protocols keep deadline fulfillment.

![Figure 6: Mode change times](image)

Experiments to validate mode change performance during application reconfiguration have also been carried out. Figure 6 shows the time taken to perform the switching among two application quality levels by exchanging the actual current task set. In the presence of high interference, results show that mode change has an average time penalty of approximately 39 ms, being its maximum value 42.3 ms.

Results for the immediate protocol shows a high stability. Peaks observed in the experiments correspond to the initialization of the system and buffers (since the results shown present execution from time 0) and to the TimeSys virtual machine periodic timers.

8. Conclusions

Predictable and QoS-based resource management in HQMES is a complex problem that is still being solved from different point of views. Existing approaches remain at either too pessimistic level for the nature of multimedia systems or leave predictability out the picture (as most general purpose traffic-based QoS scheduling only concentrates on the network resource). This paper has addressed two of the different sides of the problem: the complete characterization of pipeline-based multimedia applications to be appropriately managed by a centralized entity such as a QoSRM, and the elaboration of reconfiguration algorithms for two application types. These techniques have been integrated in the context of a contract model and in a component-based and modular architecture harness for QoS management based on HOLA-QoS; validation results have been presented that show the feasibility of the approach.

8. Bibliography


