Available online at www.sciencedirect.com
SCIENCE DIRECT®
Signal Processing: *Image Communication* 21 (2006) 200–216
SIGNAL PROCESSING: IMAGE COMMUNICATION
www.elsevier.com/locate/image

Multimedia adaptation in end-user terminals

Gianluca Di Cagno*, Cyril Concolato, Jean Claude Dufourd

Ecole Nationale Supérieure des Télécommunications, Paris

Received 12 November 2004; received in revised form 6 September 2005; accepted 28 September 2005

Abstract

In the last few years, considerable research efforts have been spent on the concept of adaptation at various points in the digital information distribution chain, from content/service generation to end-user terminals. The purpose of these efforts is to realize the Universal Media Access (UMA) vision, which means access to content by any terminal and any network. The quality of the user experience then depends on the effectiveness of the multimedia customization process. Complementary to these efforts, this paper addresses the problem of adapting the playback of a multimedia application in multimedia end-user terminals by (1) showing its relationship with MPEG-21 concepts, (2) framing it in the context of terminal Quality of Service (QoS), (3) presenting a complete framework based on the MPEG-4 systems' architecture and (4) finally presenting some results.

Keywords: Multimedia adaptation; Usage environment description; MPEG-4; MPEG-21; QoS

1. Introduction

The problem of adapting the digital media content to different networks, terminals and users has been—and is currently being—investigated by many researchers. The purpose of these efforts is to realize the Universal Media Access (UMA) vision, which means access to content by any terminal and any network. Towards this target, media adaptation might be performed at different entities of the digital information distribution chain. Media servers for example can adapt the content to different client capabilities (i.e. display size, type of CPU), once these are properly described. Similarly, intelligent gateways can adapt the content to different network characteristics, when the properties of the content (i.e. type of media, encoding format) and networks are again properly described. Many tools have been standardized in order to achieve system interoperability by the ISO/IEC JTC1/SC29/WG11 working group (MPEG). Media coding is addressed by the MPEG-1-2 and 4 standards [11], content description is addressed by the MPEG-7 standard [28] and usage environment (terminal, network, user preferences) is addressed by the MPEG-21 standard [21], while specifying a framework for multimedia delivery and consumption that aims at realizing the UMA vision.

Multimedia customization using these concepts is illustrated in detail in [18] together with an experimental architecture. The same paper also mentions that multimedia customization in end-user terminals is difficult because of the high overhead of the processing algorithms. Adaptation

*Corresponding author.
E-mail address: gianluca_dicagno@yahoo.com (G. Di Cagno).

0923-5965/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2005.09.005

engines are hence preferably located in entities different from end terminals. The assumption in this vision is then that a terminal always receives the "perfect" content, i.e. one that is adapted to user, terminal, and network characteristics. If variations in these occur during the playback, then upstream communication between the client and the server can re-start the adaptation process so that again a terminal can process content adapted to new usage characteristics. In this paper, we establish a framework for multimedia adaptation in end terminals that addresses the cases where upstream communication for adaptation is not possible because of the service scenario or because of the timescales of the terminal resource variations. In broadcast scenarios, for instance, even if the content is designed to comply with the characteristics of terminals, some variations in the computational power of end terminals may occur, due for instance to power management schemes. As a consequence, end systems should nevertheless be able to perform some adaptations of the content that they process, varying as a result their computational requirements. Besides, even in scenarios where client–server upstream communication is possible, adaptation in entities different from end terminals is not always the best option. Terminals running on general-purpose operating systems for instance show a high variation in the resource availability due to competition among applications. Then the timescales of these variations are not compatible with a communication adaptation schema. As an example, suppose that there exists a multimedia application composed of a certain number of concurrent video streams. We suppose that the multimedia customization architecture has determined the best viewing option for each media, for a given terminal, before the start of the playback of the scene. After say 20 min of playback the available CPU power of the system decreases, for instance because of other processes running on the same system. The scene cannot be supported optimally on the system in the new configuration. An intelligent multimedia terminal should be able to detect this change and change the quality options of the media that constitute the presentation accordingly. Since the variation is temporary and has timescales of hundreds of milliseconds, communicating with the server is not the best option. So eventually the terminal can degrade the rate of the video presentation or the quality of the video decoding of some streams, trading the quality of the playback with the availability of resources. Even if this problem falls into the scope of retaining a terminal Quality of Service (QoS) by taking proper management actions when necessary, we observe that there are some synergies between multimedia customization and terminal QoS for what concerns the descriptors that both mechanisms need in order to achieve their goals.

In this paper we establish a framework for adaptation at multimedia end terminals, identifying a set of user-level QoS descriptors that can be used for this purpose and we show how they can be expressed in a standardized way using MPEG-21 tools for digital item adaptation. We then complement the MPEG-21 terminal QoS model in defining the mapping between user- and application-level QoS requirements, and proposing a simple adaptation engine that selects the quality levels of each media of a presentation according to the computational requirements of each level. The framework has been conceived to provide quality management of multimedia applications running on general-purpose operating systems. Even if other frameworks for quality management are reported in literature [9,22], our contribution differs since we make no assumption on support from the operating system and we re-use information provided by the customization process in order to provide adaptation in end-terminals. We illustrate the framework considering MPEG-4 System 2D terminals presenting multiple streams concurrently, with graphic enhancements. We proceed as follows: Section 2 presents the terminal model and the assumptions we made for the purpose of this work, Section 3 presents the framework architecture, Section 4 illustrates some experimental results and finally Section 5 draws the conclusions of this work.

2. End-user terminal model

The problem of designing terminals capable of adapting the playback of multimedia content to varying system conditions can be framed in the context of QoS. In fact, when defining and implementing a terminal QoS model, the quality of the playback is constantly monitored, violations of the requirements are detected and recovering actions are initiated. In this context adaptations are seen as a result of recovering actions. Important contributions in the field can be found in the work of Campbell [4–6]. The papers clarify the need for an end-to-end infrastructure for QoS in distributed

multimedia systems proposing a layered architecture of a system with QoS mechanisms present at each layer. Our contribution focuses on end systems, and on the problem of adapting the quality of the playback to the available computational power. So we do not consider network bandwidth issues, and we assume that the data a terminal need are always available when requested. The consequence of our assumption is that in the terminal QoS layered model we consider *user*- and *application*-level issues. User-level QoS specification deals with the concept of quality as perceived by the user. Application-level QoS specification deals with the identification of QoS parameters and the specification of control processes for the system components of a terminal. In the following, we illustrate the terminal QoS model focusing on the user- and application-level issues.

2.1. User-level QoS

In defining a user-level QoS specification, we model a multimedia terminal as a *user-centric* system. We assume that users provide the notion of value of the different media of a presentation to a terminal, and using this notion an adaptation engine located in the terminal can modify the presentation of a multimedia scene under unpredictable events. A user-level quality specification then determines how the different components of a presentation contribute to the value perceived by the user. In the process of defining the specification we borrowed some concepts developed in the field of operating system schedulers, where the use of value functions to maximize system utility has been investigated. For instance in [14,15], the authors report on the use of discrete quality dimensions to maximize task allocation in an OS scheduler. The assumption in these works is that the user is able to identify a number of desirable quality dimensions for an application and their associated quality options and, most importantly, rate them. Since a multimedia scene is potentially composed of several streams, a global quality specification would not best capture the needs of the application. Consider for instance the MPEG-4 Systems application reported in Fig. 1. It shows a kart racing application that provides different views of a sport event.

Three small QCIF (176×144 pixels) videos (10 fps) provide different views of the race and one CIF (352×288 pixels) video (30 fps) provides a better-quality view. The author can certainly

Fig. 1. Kart racing application.

identify a set of qualities that may express the concepts of timeliness (frame rate), size, or perception (post-processing filters, antialiasing) for the presentation of this event. Clearly, if these options are declared per stream, the terminal can estimate the resources needed for a quality option and assign the most appropriate level of quality to each stream. As an example, suppose that during the playback of the kart application the frame rate of the CIF video suddenly decreases, because the CPU is overloaded. The CIF video is probably not the best candidate for quality degradation, because it is the most important view for the user. Instead, the three preview QCIF could be degraded (reducing the frame rate or substituting the videos with some text), and the released resources could be used to display the CIF video at full quality. In order to achieve this, quality options should be declared per stream, and each stream should convey a notion of importance relative to the others. Since multiple quality dimensions may be associated with each stream (ex: frame rate options, frame size options, visual quality options), there is also a need to rate each dimension, in order to choose among possible options taking into account the value that each user assigns to each dimension. For instance the quality dimension "size" of a video stream is a very important quality parameter for an action movie, while for a TV news programme it is not as important. Similarly, the size of the screen is very important for remote surveillance applications, which, on the other hand, might tolerate frame rate reduction.

A similar concept applies to 2D graphics. Fig. 2 shows a video catalogue application. Six small

Fig. 2. Video catalogue application.

QCIF videos scroll from right to left on the lower part of the screen. When the user clicks on one video, a larger view is displayed on the upper part of the screen. Scene description quality dimensions may include viewport (display) sizes and frame rates for the animation. If the user assigns a value to each dimension, when the presentation system assigns quality options to each stream it may trade off resource availability and user preferences. The above examples suggest the structure of a user-level QoS specification: each stream should convey its list of QoS dimensions, and their contribution to user satisfaction. The relative importance of each stream should also be evident by the description. According to this, a user-level specification of the application shown in Fig. 2 might be:

In the above example, a user-level specification is associated with each stream of the application. This specification is created at authoring time and then conveyed to the terminal in the way described hereafter. It is composed of a list of quality dimensions declared for each stream. In the example the quality dimensions capture timeliness, size and visual qualities, but the list is not meant to be exhaustive. Each quality dimension contains a list of ordered quality options, in descending quality order. Each dimension has an associated dimension value, which establishes a ranking among the dimensions in the user satisfaction. For instance in stream 1 the user rates Visual Quality as the highest quality to preserve, followed by frame rate and finally frame size. This means that the first action that a terminal would consider when degrading the quality of the scene is a reduction in frame size. Each stream also has an associated *Stream Value*. This establishes a ranking among the streams of a presentation. Streams with smaller values are the first ones to be degraded.

While we can easily extend the MPEG-4 Systems QoS model with new descriptors that carry this specification [8], it is interesting to see how we can express this specification using *standardized* tools of MPEG-21. In the MPEG-21 framework users exchange, consume and manipulate Digital Items (DIs). A DI is composed of resources, with their associated description and rights. MPEG-21 parts of the standard hence include the declaration of Digital Items (DID), the identification (DII) and, in order to realize the UMA vision, the adaptation (DIA). The target of DIA is to provide tools to support the adaptation of resources and descriptions to various terminal network and user characteristics. So DIA is comprised of *Usage Environment*, *Digital Item Resource Adaptation* and *Digital Item Declaration Adaptation* tools. *Usage Environment* specifies user characteristics, terminal capabilities, network capabilities and natural environment characteristics. The *User Characteristics, Presentation Priority Preference* tool can be used to express the difference in the importance of the media that compose a presentation. This tool can be used by an adaptation engine outside the terminal to customize the content according to user preferences; it can also be used by an adaptation engine located in the terminal to transitorily adapt the presentation to new system conditions according to presentation preferences. We can use this tool to express the *stream value* field of the user-level QoS specification, establishing a ranking among the different streams of a presentation. The *Digital Item Resource Adaptation, Terminal and Network QoS* part of the standard provides the *Adaptation QoS* tool, whose *UtilityFunction* module can be used to express the relationship between constraints (i.e. CPU, bandwidth), *AdaptationOperators* (i.e. coefficients pruning, frame skipping) and *utilities*, that is, the value of adapted content as perceived by the user. We mapped the user-level specification of stream 1 in Table 1 in the manner show in Table 2. We used the AdaptationQoS tool and its UtilityFunction module. Since we consider only computational issues, in Table 2 the constraint is the CPU and its values express the estimated percentage of CPU for a given combination of adaptation operator values. In order to map our quality dimensions we introduce new Adaptation Operators (Frame Rate, Frame

Table 1
User-level QoS specification for the scene in Fig. 2.

```
Stream 1: (Video CIF)
 Visual Quality {HIGH_QUALITY, LOW_QUALITY}, Dimension Value: 3
 Frame Rate {25, 12, 5}, Dimension Value: 2
 Frame Size {CIF, QCIF}, Dimension Value: 1
 Stream Value {3}
Streams 2, 3, 4: (Video QCIF)
 Visual Quality {HIGH_QUALITY, LOW_QUALITY}, Dimension Value: 2
 Frame Rate {10, 5, 2}, Dimension Value: 1
 Frame Size {QCIF}, Dimension Value: 3
 Stream Value {2}
Stream 5: (Audio)
 Audio Quality {HIGH_QUALITY, LOW_QUALITY}, Dimension Value: 1
 Audio Sampling Rate {44100, 22050}, Dimension Value: 2
 Stream Value {4}
Stream 6: (BIFS)
 Visual Quality {HIGH_QUALITY, LOW_QUALITY}, Dimension Value: 1
 Frame Rate {10, 5, 1}, Dimension Value: 2
 Frame Size {100, 50, 10}, Dimension Value: 3
 Stream Value {1}
```

Table 2
User-level QoS specification using DIA tools

```
<DIA>
</DescriptionMetadata>
<Description xsi:type = ''AdaptationQoSType''>
        <Module xsi:type = ''UtilityFunctionType''>
            <Constraint iOPinRef = ''CPU''>
              <Values xsi:type = ''IntegerVectorType''>
                <Vector>40 30 20 10 5 1</Vector>
              </Values
            </Constraint>
            <AdaptationOperator iOPinRef = ''FRAME_RATE''>
              <Values xsi:type = ''IntegerVectorType''>
                <Vector>25 25 25 12 5 5</Vector>
              </Values>
            </AdaptationOperator>
            <AdaptationOperator iOPinRef = ''FRAME_SIZE''>
              <Values xsi:type = ''IntegerVectorType''>
                <Vector>100 50 25 25 25 25</Vector>
              </Values>
            </AdaptationOperator>
            <AdaptationOperator iOPinRef = ''VISUAL_QUALITY''>
              <Values xsi:type = ''IntegerVectorType''>
                <Vector>1 1 1 1 1 0 </Vector>
              </Values>
            </AdaptationOperator>
            <UtilityRank iOPinRef = ''RANK''>
              <Values xsi:type = ''IntegerVectorType''>
                <Vector>6 5 4 3 2 1</Vector>
              </Values>
            </Utility>
        </Module>
</Description>
</DIA>
```

Size, Visual Quality). Quality options for each quality dimension are mapped to the *values* of each Adaptation Operator. We express the value of each combination using the *UtilityRank* field that expresses a value for each combination of adaptation operator values.

It should be noted that the use of the CPU constraint is expected to express a value for a given combination of quality options; however, the terminal can have tools to better measure complexity for a given combination of quality options. Even in this case the specification preserves its importance in providing the set of available quality options and their associated value as perceived by the user.

As we observed, the user QoS specification we illustrated in this section can be expressed using tools for Digital Item Adaptation defined in MPEG-21. In the rest of this paper we will continue to use the terminology we introduced in Table 1, comprised of quality dimensions and quality options, since in the QoS literature this is used more, and, as we will see in Section 3.2, combinatorial problems linked to the quality management process have been defined using a similar terminology. The mapping between the two representations, however, can be done in the same manner as in Table 2, for the other examples reported in this work.

2.2. Application-level QoS

In the process of defining application-level QoS we considered the system components of a multimedia terminal that complies with an international standard, MPEG-4 Systems [11]. MPEG-4 Systems specify a multimedia terminal capable of decoding and synchronizing different compressed media (''elementary streams''), composing them according to a scene description stream (BIFS) in a 2D/3D application. The system components of such a terminal are rather generic and common to other multimedia systems. As we mentioned above, application-level QoS concerns the identification of parameters that capture the quality of the playback and the specification of control processes for the system components of a terminal. In a 2D MPEG-4 Systems terminal the presentation system typically performs the display at some frame rate traversing the scene graph and composing video and audio units from different media with 2D synthetic graphics. The quality of the presentation process can be defined in terms of the rate of the output of each media and measures of the synchronization among media. Several contributions have been reported by researchers in this field. In [2] the authors provide a complete characterization of multimedia flows, defining the concepts of inter-stream, intra-stream synchronization and illustrating every possible relationship among media streams in a multimedia application. Intra-stream QoS metrics for continuous media try to quantify the *continuity* of the flow of media streams. In our framework we identified:

- *Average frame rate*: it captures the rate at which the presentation units of a stream are displayed. A nominal frame rate is selected as QoS baseline according to the available resources and user preferences.
- *Frame rate variation*: this indicates the allowable rate change in the display of presentation units. This metric depends mainly on the capabilities of the terminal and the service scenario. If the maximal frame variation is violated then the system executes management actions and computes new QoS baselines.
- *Instantaneous drift*: it indicates the delay accumulated by a decoding/rendering process. It is defined as the difference between the current time of the clock of the stream and the time stamp of the unit being processed.

It should be noted that performance metrics are dependent on the presentation algorithm and the service scenario. In networks where constant end-to-end delay is not guaranteed, other metrics like the coefficient of output frame variation may be added [10], in order to obtain a measure of the smoothness of the output units. However, for the scope of this work the above metrics are sufficient to monitor the performance of the display of each stream.

As a measure of synchronization between media streams, we used what is commonly called 'skew' to indicate the time difference between related presentation units from different streams. Steinmetz [26] reported a series of experiments a human media perception, which may be used as QoS guidelines to assess the quality of synchronization. The results of his work show that streams that have skew values greater than zero are still perceived as in sync until the skew reaches precise experimental values. This means that some misalignments can be tolerated without degrading the quality of the presentation. Using these results we derived tools to assess the

quality of the synchronization achieved by the presentation system. In the next section we illustrate the control processes and the quality management process, while introducing the terminal architecture.

3. Terminal architecture

The terminal architecture that we propose is an extension of the MPEG-4 system architecture. The architecture extensions are meant to provide support for the control of the playback of a multimedia scene according to a user-provided quality specification. In the design phase we addressed the requirements investigated in control theory for discrete event systems [12], where the issues of controllability, robustness and reactive configuration have been identified. Even if a multimedia terminal can be defined as a time-driven system (at every clock tick events occur that advance system behaviour), the interactions with a hosting environment are rather event-driven (events not known in advance and not necessarily coinciding with clock ticks interfere with the behaviour of the terminal). *Controllability* refers to the capability of a system to perform according to the specifications when input load changes dynamically (between specified bounds). This is particularly important for a multimedia terminal whose data load may change over time. *Robustness* deals with unanticipated variations in the environment of a system. As the environment departs from the expected one, the system degrades gradually in performance rather than showing a catastrophic failure. This concept is relevant for terminals running in an uncontrolled environment (i.e. general-purpose operating systems) where applications compete for system resources. *Reactive configuration* indicates the capability of a system to reconfigure its main algorithms scaling their computational needs, and is triggered by an evaluation indicating that the system is not accomplishing its mission. Controllability and robustness have been addressed by a design pattern called distributed control, illustrated in Section 3.1. Reactive configuration has been addressed in the context of the quality management process (Section 3.2).

3.1. Distributed control

In order to address robustness and controllability, the load of a system must be constantly monitored, variations in the load identified, and a reconfiguration of the terminal data-processing algorithms can eventually take place in the context of a quality management process. A major problem in dealing with the load of a multimedia system is that it varies over multiple timescales, as also observed in [3]. More precisely:

- data changes in continuous media usually take place at a time scale of tens of milliseconds (i.e. time-varying complexity of MPEG decoding [1]);
- scene changes usually take place at a time scale of seconds (i.e. BIFS scene updates possibly add or remove streams from the presentation, which may impact the terminal load);
- user changes have time scales of minutes (i.e. user interaction may trigger events that cause load variations); and
- environment changes have time scales of hundreds of milliseconds (i.e. load variations due to other applications running on the same OS).

In order to cope with these requirements, a hierarchical control architecture has been conceived. In the theory of hierarchical control systems [27], separate control units make independent observations and have their own a priori information and control variables. Control is achieved via collaboration among independent units. A control unit is constituted of an observation process that collects information about the controlled system and its environment and a decision process which uses this information and any a priori information to perform the control. Following these design principles, we conceived the terminal architecture depicted in Fig. 3.

The architecture is an extension of the MPEG Systems' one [16], since the main concepts of having separate decoding pipelines, a scene graph data structure, and a Presenter component that performs the rendering of the nodes of scene graph, accessing composition units produced by decoders, are also present in the proposed architecture. Here we addressed the control at different time scales, achieved with the collaboration of the different control units located in the Decoders, in the Presenter component, and a new component called Resource Manager. It should be noted that even if the need for control units and a Resource Manager is a concept already reported in other contributions [9], in our approach these are components to be included in each application rather than centralized as components on top of the operating system. This is because we do not consider specific real-time

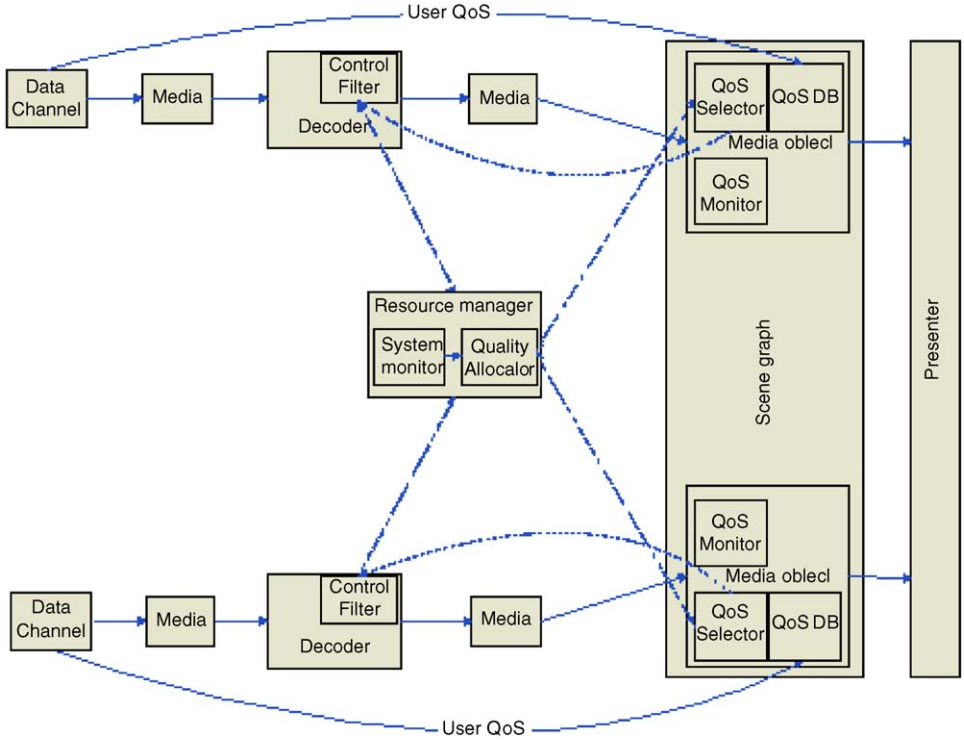

Fig. 3. Terminal architecture.

systems as in [9] but general-purpose operating systems and platforms where applications have to be able to adapt independently. In the following we briefly describe these components and their interactions.

- The decoder's control units monitor dynamic load variations of continuous media, and the progress of the decoder process. These units observe the incoming data of media decoders, and perform media-dependent control. Control on media decoding is typically performed on video decoders. Two main strategies exist: quality reduction and frame skipping. Quality reduction is inspired by the imprecise computations model [17]. If a video decoder provides some form of downgraded decoding algorithm (i.e. complexity scalability via IDCT data pruning [23]) control units may predict the complexity of incoming data, using either complexity measures carried in the MPEG-4 Visual syntax [19] or performing an estimation extracting parameters from bitstream headers [1,13] and regulate the decoding process to produce frames of inferior quality if the complexity of a frame is superior to a threshold selected by the Resource Manager. A simpler strategy is frame skipping, where control units regulate the decoding process forcing the decoder to skip part of the incoming data.
- Presenter control units monitor the continuity of the flow of a media accessing information associated with each media node ("QoS monitor" in Fig. 3). For instance if we consider video streams, whenever the associated media node has a new frame to display, the current frame rate f_c is calculated. If f_r is the nominal frame rate selected by the current QoS baseline and r_c is the max rate variation, then f_c has to be between f_r and $f_r^*(1-r_c)$. If this condition is over looked a notification is sent to the Resource Manager. Scene graph changes (caused by user interaction or scene updates) are monitored using information encapsulated in the root of the scene graph. Control units for audio streams observe the consumption of audio samples and monitor the progress of the stream, comparing the time stamps of the samples being played with the stream time base clock. When a deviation is detected and is superior to a tolerable skew, then proper action is taken: samples are inserted or the clock is updated. This control is meant to compensate potential drift between the audio

board clock and the system clock. Control on the audio clock is also meant to provide synchronization between streams. Several algorithms, to achieve synchronization are available in the synchronization literature [10] and the MPEG systems' literature (master/slave technique reported in [24]). The QoS synchronization specification for MPEG-4 terminals expresses the acceptable violations of the MPEG-4 Systems Decoder Model. The System Decoder Model provides an ideal definition of the behaviour of the terminal, ideal because it assumes instantaneous decoding time and constant end-to-end delay. This implies that composition units are available for presentation at the time expressed by their composition time stamp. It follows that, in the ideal model, assuming the same time base, composition units from different streams with the same time stamp have to be composed at the same instant, and any drift would be a violation of the model. However, some streams may tolerate little violations of the model. If two or more streams are supposed to be synchronized, they have to refer to the same time base, and so they share the same clock object. Intra- and interstream quality of synchronization then depends on how close the composition time stamps of the units being displayed are, with respect to the stream clock. QoS for synchronization does not require interactions between the Resource Manager and other control units, but has to be taken into account by the synchronization algorithm as proper thresholds to consider before taking corrective actions.

- The Resource Manager observes environment changes, typically CPU usage of the terminal process and other operating system processes. General-purpose operating systems are best-effort operating systems, and hence they gave no guarantees to applications. They do not reject an application during overload, but instead reduce the processor time available to other applications to "make room" for the new one [25]. Since these kinds of situations cannot be predicted, applications should monitor the usage of the resources of the terminal and be able to dynamically adapt their needs. The System Monitor component of the architecture mainly monitors CPU usage, at regular intervals. The load of the multimedia terminal process and the total load of the system are quantified. It must be noticed that resource monitoring alone is not sufficient to determine whether an application is successfully carrying out its tasks. A busy processor could mean that it is handling considerable work well, or that is overloaded. Hence, resource monitoring has to be coupled with application QoS monitoring. If QoS violations are detected and CPU is overloaded, then the quality management process will reduce the computational needs of the terminal, reassigning quality options to each stream.

As we saw, an observation performed by control units is meant to indicate QoS violations. This may lead to independent decision processes, or notifications to the Resource Manager that will eventually start recovering actions, through the quality management process (in this sense we designed a hierarchy in the control). The need for an additional, separate entity (the Resource Manager) is justified by the fact that we have to centralize the analysis of QoS violations trading off user preferences and resource availability. This is illustrated in detail in the following section.

3.2. Quality management

The process of assigning quality options to the streams of a presentation is connected to the available resources of a terminal. If a terminal had infinite resources, the best-quality options would always be selected. Since this is obviously not the case, qualities have to be assigned so that the resource usage does not exceed the available resources. Besides, since the load of a terminal changes over time, quality options have to be reassigned whenever changes in the load are predicted (if possible) or identified or when violations of the performance and synchronization requirements are detected. Analogously, the quality management process has to take into account the user-level QoS specification, that assigns different values to the streams of a presentation, and also expresses different values for the recovery actions (quality dimensions) available for each stream. This specification is meant to be provided by the author of the multimedia scene, and conveyed to the terminal as described in Section 2.1. Hence the presentation system has to select the quality options for each quality dimension of a stream in order to maximize user satisfaction, given the constraint of the available CPU power. As shown in [14], these combinatorial problems (Single Resource Multiple QoS

dimensions) are NP-hard, since they can be considered as an instance of the 0–1 Knapsack Problem. In our framework, we adopted a simple greedy algorithm to choose qualities based on the stream value and the resource consumption associated with each quality level. We chose a greedy algorithm on the basis of the following considerations:

(1) In a 2D application, the number of streams and quality dimensions is not expected to be so high to justify a more complex algorithm. Note: the framework assigns qualities to streams. 3D frameworks using similar approaches assign qualities to each object in a 3D world; in this case more complex algorithms are necessary since the number of objects and the number of quality options per object can be quite high [22].
(2) The framework assigns the same quality options to streams with the same stream value. This is because streams with the same stream values are considered to share some semantic meaning. Quality options are hence assigned so that streams with the same semantic value share the same quality options. This also simplifies the algorithm.

We observe that point (2) is not a limitation, since if we need different quality options to be allocated to different streams it is sufficient to assign them different stream values. The greedy algorithm performs the following steps, assuming as input a value $R_{\max}$ of available CPU, and a user QoS specification as defined in Section 2:

- Order the streams in decreasing importance, grouping streams with the same stream value.
- Order quality levels in decreasing quality order. Ordered quality levels are determined by combining the quality options of each quality dimension, starting from higher dimension values to lower ones.
- For each group of streams, starting with the group at the highest stream value:
  - Start selecting the quality level that corresponds to the highest quality.
  - Estimate resources for this group at the selected quality level.
  - If the estimation is higher than $R_{\max}$, select a lower quality level and again perform an estimation until the condition is satisfied.
  - If no combination of quality level is found, select the lowest quality level and raise exception.

We observe that in order to be effective, the algorithm needs models to estimate the resource consumption of a stream at a selected quality level. In Section 4 we discuss this subject for 2D applications. In the context of 3D applications, significant contributions can be found in [22].

In the architecture of Fig. 3, quality management is performed via collaboration of the Resource Manager (Quality Allocator component) and the following components of the media nodes associated with each stream in a presentation: Quality DB and Quality Selector. The Quality Allocator runs the quality management algorithm illustrated above in order to assign qualities to each stream. The Quality DB of each media node contains the stream user QoS specification specified in Section 2. The Quality Selector of each media node is responsible for setting the presentation of each stream to the combination of quality options identified by the Quality Allocator.

4. Experimental results

In the following, we present some examples of utilization of the proposed framework for adaptation in end terminals. The experiments provide evidence of the use of simple adaptation engines in a multimedia terminal to perform adaptations after the multimedia customization process. As such, in these examples we assume that the scene that the multimedia terminal received has already been customized, and we describe situations where the terminal has to adapt its running conditions. In particular, in the first example we show how the proposed framework can be used to adapt the playback of an application comprised of multiple video streams, when there are variations in the available power due to competition among system processes and when there are variations due to power management schemes. This example covers two important scenarios of terminals running on general-purpose operating systems. The second example shows a quality management scenario where the presentation rates and the size of the video objects are adapted depending on the available computational power. The third example shows the use of visual quality descriptors, showing how the framework can be used to adapt the quality

of the rendering while maintaining fixed presentation rates when displaying 2D animated cartoons.

4.1. Experiment 1: adapting presentation rates of multiple video objects

Here we consider a scene constituted of 10 video streams. Ten BIFS *MovieTexture* nodes are spatially composed in a BIFS scene as illustrated in Fig. 4. The 704×576 screen is divided into four 352×288 areas, containing the following from the top left:

1 video CIF MPEG-4 Advanced Simple Profile (ASP) 1 Mb/s, 25 fps (V1);
4 video QCIF Advanced Simple Profile 250 Kb/s, 25 fps (V3–V6);
4 video QCIF Advanced Simple Profile 250 Kb/s, 25 fps (V7–V10); and
1 video CIF MPEG-4 Advanced Simple Profile 1 Mb/s, 25 fps (V2).

Fig. 4. Experiment 1, BIFS scene comprised of 2 CIF and 8 QCIF streams.

A user QoS specification for the scene in Fig. 4 may assign different values to the CIF and the QCIF videos, using the *stream value* parameter illustrated in Section 2. Three levels of interest may be expressed (in descending order): V1 and V2 (the two CIF videos) have the same highest value, V3, V4, V5, V6 share the same middle value and V7, V8, V9, V10 have separate decreasing values. Besides, the user may be willing to accept only frame rate variations, and accept the QCIF nodes to be completely switched off, if it is necessary. The complete user QoS specification is reported in Table 3.

In order to select a quality level for each stream according to the available power, the terminal has to associate quality levels with a measure of the computational power needed. From the specification in Table 3, the terminal is able to identify 3 levels of complexity for the presentation of each CIF node, and 4 levels of complexity for each QCIF node. We defined complexity as the percentage of CPU needed for the rendering and decoding of a media associated with a node at a given level of quality over a second. This quantity depends on the number of cycles necessary for video decoding and rendering. The number of cycles for video decoding is a quantity variable in time over bounds that depends on incoming data and software implementation on a particular architecture. At each quality level we first assigned an average decoding complexity and then updated the value during playback. We could have chosen worst-case cycles, but, as reported in [20], worst-case analysis is not adequate to define a useful decoding complexity. The same paper reports that variances of complexity figures for fixed bitrates do not show large variations. Hence we started with an average complexity and updated the estimate during playback. While this

Table 3
User-level QoS specification for the scene in Experiment 1

```
Streams CIF (V1 and V2)
QoS_Qualifier_FRAME_RATES {FRAME_RATES [25,8,2]}, DimensionValue 1
QoS_Qualifier_STREAM_VALUE {STREAM_VALUE 6}

Streams QCIF (V3, V4, V5, V6)
QoS_Qualifier_FRAME_RATES {FRAME_RATES[25,8,2,0]2,8,25}, DimensionValue 1
QoS_Qualifier_STREAM_VALUE {STREAM_VALUE 5}

Streams QCIF (V7, V8, V9, V10)
QoS_Qualifier_FRAME_RATES {FRAME_RATES[25,8,2,0]2,8,25}, DimensionValue 1
QoS_Qualifier_STREAM_VALUE {STREAM_VALUE 1-4}
```

approach is clearly not suitable for scheduling on real-time systems, it can be used for the purpose of this work to validate the architecture that we illustrated. Better complexity estimation figures, if available, can be retrieved from information inserted at the encoder side [19].

4.1.1. General-purpose OS, competition between applications

We first compared the playback of the experiment 1 scene in a QoS-enabled terminal and a QoS-disabled one, when other processes were running on the same terminal, inducing heavy load. We tested the scene on a terminal with a Pentium 1.3 GHz CPU and a Windows platform. In this set-up, the scene consumes from 50 to 60 per cent of the CPU and all the streams run at full quality, having frame rates of 25 fps for the whole duration of the simulation. In order to add CPU workloads at different levels, we used a tool commonly used to simulate processor workload on Windows platforms, CPU Stress. We ran the MPEG-4 player and the CPU processes and we observed the behaviour of the player. In performing the simulation, we took care in avoiding any difference in the priority of the two processes. First the player was launched; then, after 12 s the CPU Stress process was activated, simulating an additional workload of 60 per cent of the CPU. Fig. 5 plots the CPU utilization as a function of the time for the CPU stress process (circles line), the MPEG-4 player (triangles line) when QoS is disabled and the utilization of the whole system (block line). As we can see, when the CPU stress process and the MPEG-4 player run concurrently, the system is overloaded, since the overall processor activity is always at 100 per cent. When both applications are running, the two processes compete for resources and the OS scheduler tries to fairly share resources between the two applications, assigning nearly 50 per cent of the CPU to each process.

Fig. 6 shows the interaction between the CPUStress tool and the MPEG–4 player when QoS is enabled.

As we can see from the picture, the MPEG-4 player with QoS enabled is "adaptive", in the sense that it decreases its resource consumption to the available power, so that the system is not constantly overloaded, and the CPUStress process can run at its normal full power. Frame rates for four of the streams of Experiment 1 are reported, for both simulations, in Figs. 7 (QoS disabled) and 8 (QoS enabled). In Fig. 8, we observe that the frame rates decrease to a value of 10–15 frames/s as soon as the

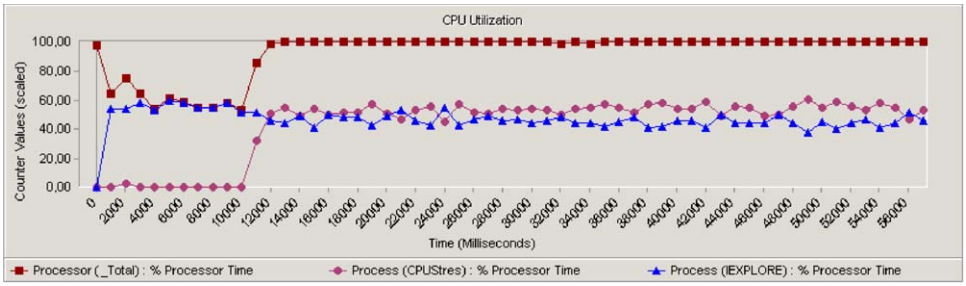

Fig. 5. CPU utilization when MPEG-4 Player and CPUStress tool are active, QoS is disabled.

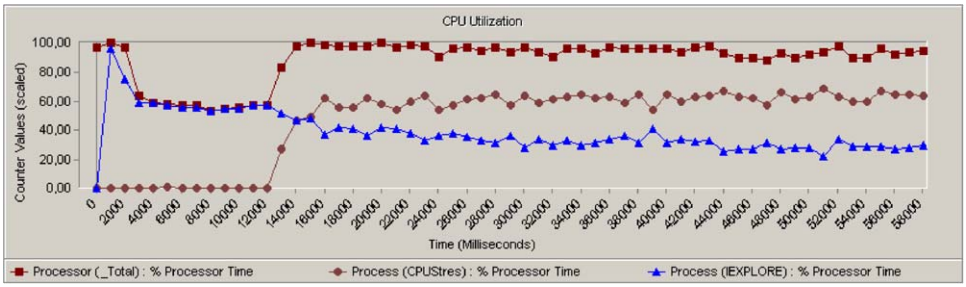

Fig. 6. CPU utilization when MPEG-4 Player and CPUStress tool are active and QoS is enabled.

CPUStress process is activated. All the streams have the same presentation rate. In Fig. 9, we observe that the quality management algorithm has selected a combination of rates according to the user QoS specification of Table 3. It has to be noticed that, when QoS is enabled, stream 1 and stream 3 show some variations in the frame rates. This is because the configuration of this example imposes a high stress on the system, and a high variation in the amount of computational resources available. Nevertheless, the variation in the frame rate occurs between the tolerable thresholds set by the QoS specification.

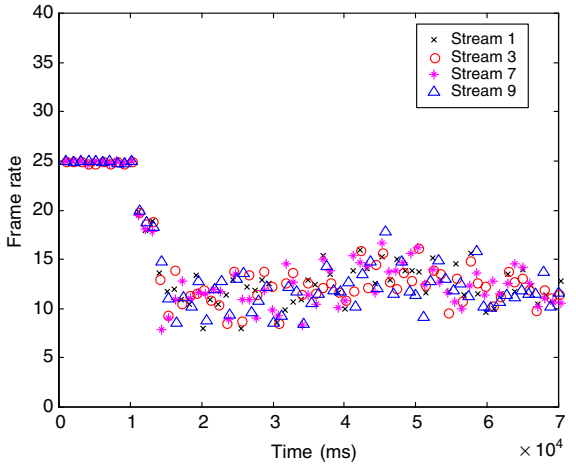

Fig. 7. Frame rates when MPEG-4 Player and CPUStress tool are active and QoS is disabled.

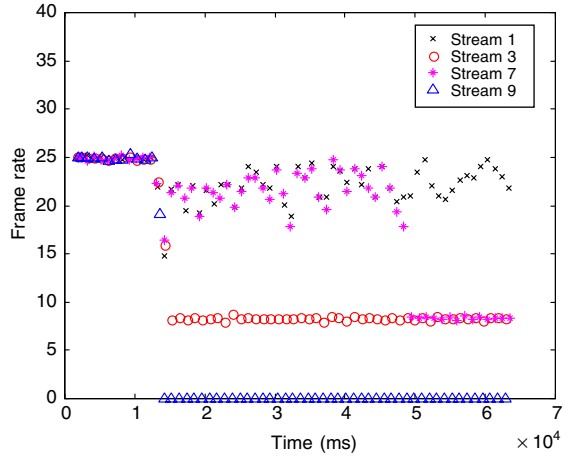

Fig. 8. Frame rates when MPEG-4 Player and CPUStress tool are active and QoS is enabled.

4.1.2. *Mobile computing*

Modern portable computers can be configured to vary the CPU clock frequency when batteries are low or when there is a switch in the power source (i.e. from AC adapter to battery). Applications running on these systems should have to be able to adapt their computational requirements when such changes occur. Again we compared the playback of the scene on a QoS-enabled terminal and a QoS-disabled one, configuring an Intel Centrino laptop so that the speed of the CPU clock would vary from 1.3 GHz to 600 MHz when switching from AC adapter to battery power source. Figs. 9 and 10 show the CPU utilization during a run of 30 s. The MPEG-4 player is started in AC power mode; then, after about 10 s the AC cable is disconnected. The figures show CPU utilization of the system (square line), the MPEG-4 player (triangle line), and clock frequency (circle line) when QoS is disabled (Fig. 9) and enabled (Fig. 10). We observe that in both experiments the scene is well supported during the first 10 s, when the system is running at full power. However, when the clock frequency changes, the QoS-disabled and QoS-enabled players show different behaviours. In the first case, the system is overloaded. When QoS is enabled, instead, the overall CPU utilization is high, but the system is not overloaded. The QoS-enabled terminal reallocates resources when it detects that the scene cannot run

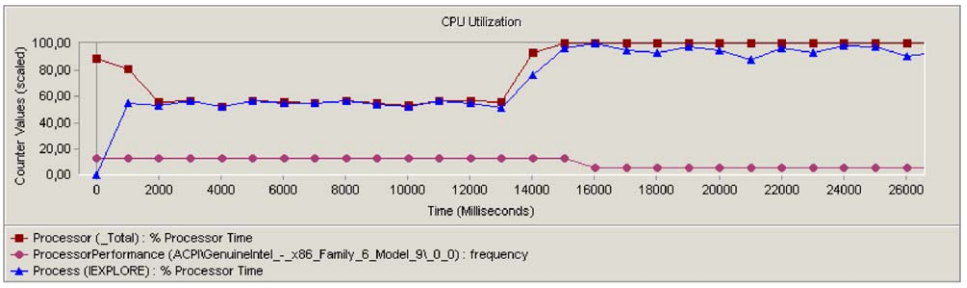

Fig. 9. CPU utilization and CPU clock frequency, QoS is disabled.

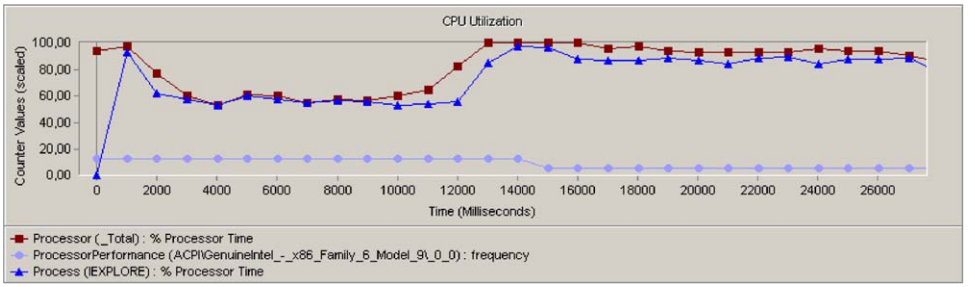

Fig. 10. CPU utilization and CPU clock frequency, QoS is enabled.

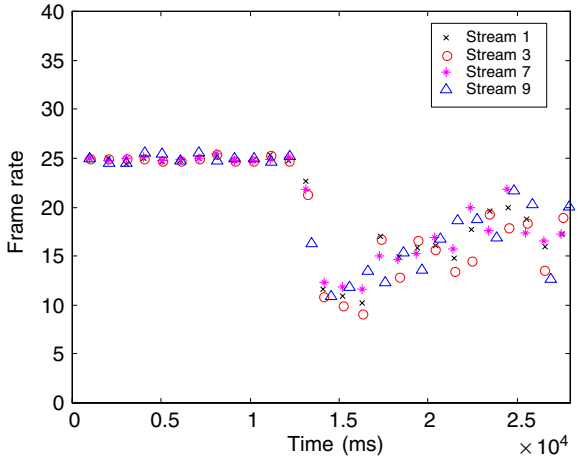

Fig. 11. Frame rates when CPU clock frequency changes, QoS is disabled.

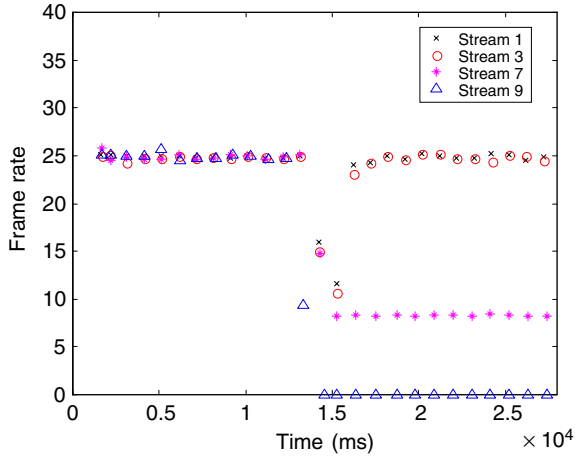

Fig. 12. Frame rates when CPU clock frequency changes, QoS is enabled.

at full quality at a lower CPU frequency. Stream frame rates (Figs. 11 and 12) are much different in the two cases. When QoS is enabled, as soon as the clock speed is slow down, the player re-assigns stream rates according to the new computational power. When QoS is disabled instead, the frame rates of all the streams decrease uniformly, as if the streams would render the same value to the user.

Fig. 13. Experiment 2 scene, full-quality view.

4.2. Experiment 2: adapting size and frame rates of visual objects

A more complex scenario of quality management is shown in Fig. 13. We considered a BIFS scene with a CIF video scaled to 704×576, with height QCIF video superimposed on screen. The QoS specification (Table 4) is meant to assign more value to streams on the left of the display, and then streams on the right and finally the scaled CIF video. In Table 4, streams V5–V8 and V9 are declared to accept variations in frame rate and frame size.

In the case of stream V9, the Frame Size *dimensionValue* is smaller than the Frame Rate *dimensionValue*. This means that the author is

Table 4
User QoS specification for the scene in Experiment 2

```
Streams QCIF (V1, V2, V3, V4)
QoS_Qualifier_FRAME_RATES{FRAME_RATES[25,8,2]}, DimensionValue 1
QoS_Qualifier_STREAM_VALUE {STREAM_VALUE 3}

Streams QCIF (V5, V6, V7, V8)
QoS_Qualifier_FRAME_RATES{FRAME_RATES[25,8,2,0]}, DimensionValue 2
QoS_Qualifier_FRAME_SIZE{FRAME_SIZE [100,50]}, DimensionValue 1
QoS_Qualifier_STREAM_VALUE {STREAM_VALUE 2}

Stream CIF (V9)
QoS_Qualifier_FRAME_RATES{FRAME_RATES[25,8,2,0]}, DimensionValue 2
QoS_Qualifier_FRAME_SIZE{FRAME_SIZE [100,50]}, DimensionValue 1
QoS_Qualifier_STREAM_VALUE {STREAM_VALUE 1}
```

Fig. 14. Experiment 2 scene, reduced-quality view.

Fig. 15. ENSTAg001b.mp4 cartoon.

willing to accept frame size reductions more "enthusiastically" than frame rate reductions. Levels of complexity are then built up in decreasing user satisfaction and stored in appropriate MovieTexture private data. As a result, in situations similar to the one reported in Experiment 1, the scene can be adapted in end terminals as shown in Figs. 13 and 14.

4.3. Experiment 3: rendering of 2D animated cartoons

We consider a scene comprised of a continuous BIFS stream and no other visual streams. The scene is conceived as a sequence of frames constituted of 2D geometric objects, providing the illusion of character animation. One frame of the scene is shown in Fig. 15.

We assume that the user is willing to accept changes in the quality of the scene instead of changes in the frame rate of the animation. The user QoS specification of the BIFS stream is reported in Table 5.

We used antialiased and normal rendering to implement the high-quality/low-quality modes for the visual quality dimension. Antialiasing rendering for lines and polygons provides a better visual quality but it requires more computational resources. Quality changes as allowed by the user QoS specification can occur during the presentation depending on the resources available (as illustrated in Experiment 1). We ran the scene first with QoS enabled, allowing the system to decide which rendering mode to use on the basis of the resources available. Then we ran the scene with QoS disabled and using high-quality, antialiasing rendering for the whole scene. Fig. 16 compares the frame rates of the two experiments, for 250 animation frames. The figure also shows the quality mode used during the rendering, for the QoS-enabled experiment. The quality level is set to 1 if the frame is rendered in low quality and 0 if a frame is rendered in high quality. We observe that, when QoS is enabled ('*' line,

Table 5
User QoS specification for ENSTAg001b.mp4 cartoon

```
QoS_Qualifier_FRAME_RATES{FRAME_RATES [12,8,2]}, DimensionValue 2
QoS_Qualifier_VISUAL_QUALITY{QUALITY[HIGH,LOW]}, DimensionValue 1
QoS_Qualifier_STREAM_VALUE {STREAM_VALUE 1}
```

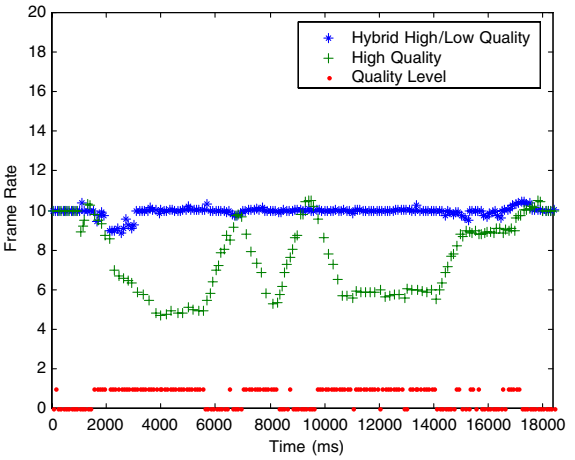

Fig. 16. Comparison of frame rates for ENSTAg001b.mp4, hybrid and high-quality rendering modes.

"hybrid high/low-quality" rendering) the terminal switches to low-quality mode in order to keep the frame rate constant from time 2000 to 4000 and from time 10,000 to 14,000, while when QoS is disabled ('+' line, "high-quality" rendering) the terminal shows low frame rates in the same interval.

5. Conclusions

In this paper we established a framework for adaptation at multimedia end terminals, identifying a set of user-level QoS descriptors that can be used for this purpose and we showed how they could be expressed in a standardized way using MPEG-21 tools for digital item adaptation. We then complemented the MPEG-21 terminal QoS model in defining the mapping between user- and application-level QoS requirements, and proposing a terminal architecture based on the MPEG-4 Systems standard. A simple adaptation engine that selects the quality levels of each media of a presentation according to the computational requirements of each level is described, and experimental results show how the framework can be used to provide quality management of multimedia applications running on general-purpose operating systems. We believe that the main contribution of this work is the framing of the problem of adapting the playback of a multimedia application to varying system conditions in the context of terminal QoS, showing the relationship between Digital Item Adaptation as conceived by the MPEG-21 standard and terminal QoS models. As a sequel to this work, we are currently investigating the integration of our framework in an end-to-end QoS architecture for the delivery of audio–visual services over heterogeneous networks, based on the MPEG-21 standard. This would leverage the results of the work, since it would consider trade-offs between computational and bandwidth issues, introducing Multiple Resource, Multiple QoS Dimension problems.

References

[1] A.C. Bavier, A.B. Montz, L. Peterson, Predicting MPEG Execution Times, SIGMETRICS '98. Proceedings of the Joint International Conference on Measurement and Modelling of Computer Systems, 1998.
[2] G. Blakowski, R. Steinmetz, A media synchronization survey: reference model, specification, and case studies, IEEE JSAC 14 (1) (January 1996) 1–2.
[3] R.J. Bril, C. Hentschel, E.F.M. Steffens, M. Gabrani, G.C. van Loo, J.H.A. Gelissen, Multimedia QoS in consumer terminals, Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS), September 2001, pp. 332–343.
[4] A. Campbell, C. Aurrecoechea, L. Hauw, A review of QoS Architectures, Proceedings of the Fourth International Workshop on Quality of Service, Paris, March 1996.
[5] A. Campbell, G. Coulson, F. Garcia, D. Hutchinson, H. Leopold, Integrated quality of service for multimedia communications, In COMM '92, 1992, pp. 99–110.
[6] A. Campbell, G. Coulson, D. Hutchinson, Supporting adaptive flows in a quality of service architecture, Multimedia Systems J. (November 1995).
[8] G. Di Cagno, Quality-Based Multimedia Systems, Ph.D. Thesis, ENST Paris, July 2004.
[9] C. Hentschel, et al., Video quality of service for consumer terminals—a novel system for programmable components, IEEE Trans. Consumer Electron. 49 (4) (November 2003) 1367–1377.
[10] Y. Ishibashi, S. Tasaka, T. Iwana, Adaptive QoS control for voice and video traffic in networked virtual environments, Computer Communications and Networks, Proceedings of Ninth International Conference, 2000.

[11] ISO/IEC 14496-1:2001. Coding of Audio-Visual Objects-Part 1: Systems, Second ed., 2001.
[12] M.M. Kokar, K. Baclawski, Y.A. Eracar, Control theory-based foundations of self-controlling software, IEEE Intell. Systems 14 (3) (May/June 1999) 37–45.
[13] C.J. Lan, Y. Chen, Z. Zhong, Mpeg2 decoding complexity regulation for a media processor, Proceedings of the 4th IEEE Workshop on Multimedia Signal Processing (MMSP), Cannes, France, October 2001, pp. 193–198.
[14] C. Lee, J. Lehoczky, R. (Raj) Rajkumar, D. Siewiorek, On quality of service optimization with discrete QoS options.
[15] C. Lee, D. Siewiorek, An approach for quality of service management, Technical Report CMU-CS-98-165, Computer Science Department, CMU, October 1998.
[16] Z. Lifshitz, G. Di Cagno, M. Leditschke, Implementing the Standard: the Reference Software, in: F. Pereira, T. Ebrahimi (Eds.), The MPEG-4 BOOK, IMSC Multimedia Series, Prentice-Hall, Englewood Cliffs, NJ, 2002.
[17] J.W.S. Liu, K.J. Lin, W.K. Shih, A.C. Yu, J.Y. Chung, W. Zhao, Algorithms for scheduling imprecise computations, IEEE Comput. 24 (5) (May 1991) 58–68.
[18] J. Magalhaes, F. Pereira, Using MPEG standards for multimedia customization, Signal Process. Image Commun. 19 (2004) 437–456.
[19] M. Mattavelli, S. Brunetton, Controlling decoding power by graceful degradation techniques, ISO/IEC JTC1/SC29/WG11/MPEG97/M1445, Maceio, November 1996.
[20] M. Mattavelli, S. Brunetton, Results of core experiment on CGD: methods to measure bitstream video decoding complexity for CGD implementation, ISO/IEC JTC1/SC29/WG11/MPEG97/M1445 Stockholm, July 1997.
[21] MPEG Requirements Group, MPEG-21 Multimedia framework, Part 1: Vision, technologies and strategy, Proposed Draft Technical Report, second ed., Doc. ISO/MPEG N6269, MPEG Waikaloa Meeting, USA, December 2003.
[22] N.P. Ngoc, W. Van Raemdonck, G. Lafruit, G. Deconinck, R. Lauwereins, A QoS Framework for Interactive 3D Applications, Proceedings of the 10th International Conference on Computer Graphics and Visualization'2002, WSCG'2002, February 4–8, 2002.
[23] S. Peng, Complexity Scalable Video Decoding Via IDCT Data Pruning, ICCE 2001.
[24] P.V. Rangan, S.S. Kumar, S. Rajan, Continuity and synchronization in MPEG, IEEE J. Select. Areas Commun. Special Issue on Multimedia Synchronization, 1996.
[25] J. Regehr, M.B. Jones, J.A. Stankovic, Operating system support for multimedia: the programming model matters, Technical Report MSR-TR-2000-89, September 2000.
[26] R. Steinmetz, Human perception of jitter and media synchronization, IEEE J. Select. Areas Commun. 14 (1) (January 1996).
[27] Z. Uykan, Hierarchical Control and Multimedia, Multimedia applications in industrial automation—Collected papers of the Spring 1997 postgraduate seminar, Helsinki University of Technology, Report 106, June 1997.
[28] P. van Beek, J.R. Smith, T. Ebrahimi, T. Suzuki, J. Askelof, Metadata-driven multimedia access, IEEE Signal Process. Mag. 20 (2) (March 2003) 40–52.