MuMiVA: a Multimedia Delivery Platform using Format-agnostic, XML-driven Content Adaptation

Davy Van Deursen*, Sarah De Bruyne*, Wim Van Lancker*,
Wesley De Neve*, Davy De Schrijver*, Hermann Hellwagner‡, and Rik Van de Walle*
*Ghent University – IBBT, ELIS – Multimedia Lab
Gaston Crommenlaan 8, bus 201, B-9050 Ledeberg-Ghent, Belgium
‡Klagenfurt University, ITEC
Universitätsstrasse 65-67, A-9020 Klagenfurt, Austria

***Abstract*— Due to the increasing heterogeneity in the current multimedia landscape, the delivery of multimedia content has become an important issue today. This heterogeneity is not only reflected by a plethora of different usage environments, but also by the presence of multiple (scalable) coding formats. Therefore, format-independent adaptation engines have to be used within a multimedia delivery platform, which are able to adapt the multimedia content according to a certain usage environment, independent of the underlying coding format of the content. By relying on automatically created textual descriptions of the high-level syntax of binary media resources, a format-independent adaptation engine can be build. MPEG-21 generic Bitstream Syntax Schema (gBS Schema) is a tool that is part of the MPEG-21 Multimedia Framework. It enables the use of generic Bitstream Syntax Descriptions (gBSDs), i.e., textual descriptions in XML, to steer the adaptation of a binary media resource, using format-independent adaptation logic. In this paper, we address the design and performance evaluation of a multimedia delivery platform that relies on gBS Schema-driven adaptation engines. This platform is called MuMiVA; it is a fully integrated, extensible platform for multimedia delivery in heterogeneous usage environments, using streaming technologies. To demonstrate the flexibility of our multimedia delivery platform, we discuss the functioning of two different applications (i.e., exploitation of temporal scalability and shot selection) applied to two different coding formats (i.e., MPEG-4 Visual and H.264/AVC).**

***Keywords*— Content adaptation, Content delivery, MPEG-21 gBS Schema, XML transformations**

I. Introduction

The efficient delivery of multimedia resources is gaining importance these days because more and more devices are able to consume multimedia content. The tremendous diversity in these devices introduces difficulties for multimedia delivery infrastructures, due to varying characteristics such as screen size, processing power, and supported coding formats. Another problem is the heterogeneity of network technologies, which differ in terms of bandwidth, jitter, and error robustness. It is clear that a transparent approach is needed in order to deliver multimedia content anywhere, at anytime, and on any device. This vision is generally known as Universal Multimedia Access (UMA, [1]).

Scalable coding is an important tool to realize a UMA environment. It enables the extraction of multiple (lower quality) versions of the same media resource without the need of a complete recoding process. The bitstream extraction process typically involves the removal of particular data blocks and the modification of the value of certain syntax elements. This approach is in line with the UMA paradigm, where a media resource only needs to be created once, after which it can be published on all possible terminals.

A multimedia delivery platform needs an efficient adaptation strategy to deal with the heterogeneity in the current and future multimedia landscape. The presence of different (scalable) coding formats requires the usage of a single, format-agnostic adaptation engine, which can be used for the adaptation of still images, audio resources, and video resources. The use of a generic adaptation engine also means that the underlying implementation does not have to be updated in case a new coding format has to be supported.

One way to realize a format-agnostic adaptation engine is to rely on automatically generated textual descriptions. These descriptions contain information about the high-level structure of a scalable bitstream and are typically expressed by making use of the eXtensible Markup Language (XML) [2].

In this paper, we introduce a fully integrated multimedia delivery platform called MuMiVA[1]. MuMiVA relies on format-agnostic adaptation engines and aims at being deployable in streaming environments. The outline of this paper is as follows. Sect. II gives a general overview of XML-driven content adaptation. The MuMiVA architecture is discussed in Sect. III. Subsequently, Sect. IV elaborates on two applications of MuMiVA, providing insight in the practical use of an XML-based content adaptation system. Sect. V provides performance measurements regarding our multimedia delivery platform. Future extensions are discussed in Sect. VI. Finally, conclusions are drawn in Sect. VII.

II. XML-driven Content Adaptation

As discussed in the introduction, XML-driven content adaptation enables the creation of a format-agnostic adaptation engine, which is important to deal with different (scalable) coding formats. Indeed, the engine operates at the XML level, and does not have to be aware of the underlying coding

[1]MuMiVA is short for "Mutare, Mittere, Videre, Audire", which is Latin for "to adapt, to send, to watch, to hear".

format. Furthermore, the use of XML descriptions allows for an integration with other metadata standards, such as Dublin Core and MPEG-7. Hence, XML-driven content adaptation can also be used to realize so-called 'intelligent' adaptations, like automatic video summarization and scene selection.

A. General Approach

An XML-driven content adaptation engine typically consists of three main processes: the generation of an XML description, the transformation of the XML description, and the creation of an adapted bitstream using the transformed XML description.

An XML description contains information about the high-level structure of a compressed bitstream. In particular, it describes how the bitstream is organized in terms of layers or packets of data. Note that such an XML description is not meant to replace the original binary data; it rather acts as an additional layer, similar to metadata.

The actual adaptation takes place in the XML domain during the transformation of the XML description (e.g., by dropping descriptions of layers or packets). This transformation process takes into account the constraints of a given usage environment (e.g., available bandwidth and screen resolution). Thanks to the use of XML, many already existing XML transformation tools can be used, such as eXtensible Stylesheet Language Transformations (XSLT, [3]) or Streaming Transformations for XML (STX, [4]).

The translation from the XML domain back to the binary domain occurs in the last step, i.e., the creation of the adapted bitstream. This process takes as input the transformed XML description and the original bitstream in order to produce an adapted bitstream, which is then suited for playback in a given usage environment.

B. MPEG-21 gBS Schema

MPEG-21 generic Bitstream Syntax Schema (gBS Schema) is a tool of Digital Item Adaptation (DIA, [5]), which is part 7 of the MPEG-21 Multimedia Framework [6]. This description tool can be used in a format-agnostic adaptation engine. Furthermore, gBS Schema enables the creation of format-independent descriptions, i.e., generic Bitstream Syntax Descriptions (gBSDs) [7]. Indeed, gBS Schema acts as a W3C's XML Schema for these gBSDs.

The functioning of a gBS Schema-based adaptation framework is illustrated in Fig. 1. In this figure, a particular scene (i.e., `scene_2`) is extracted from a video sequence by using XML-driven content adaptation. The first step is the generation of a gBSD (step (1) in Fig. 1). This process is not described in the DIA specification[2], which implies that a gBSD may be generated in any proprietary way. Subsequently, the gBSD is transformed by using common XML transformation technologies such as XSLT or STX (step (2) in Fig. 1). After the transformation of the gBSD, an adapted bitstream is obtained using the format-independent gBSDtoBin parser, which takes as input the original bitstream and the transformed

[2]Only the gBS Schema is described in the specification, together with the behaviour of a gBSDtoBin parser.

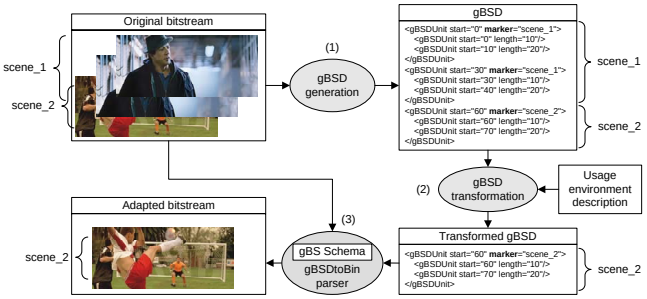


Fig. 1. Scene selection using a gBS Schema-based adaptation framework.

gBSD (step (3) in Fig. 1). The gBSDtoBin parser relies on the gBS Schema to translate the transformed gBSD into an adapted bitstream [8].

The two most important gBS Schema elements are *gBSDUnit* and *Parameter*. *gBSDUnit* represents a segment of the bitstream (e.g., a segment corresponding to a scene as illustrated in Fig. 1), while a *Parameter* is used to describe syntax elements of the bitstream that might change (e.g., a syntax element corresponding to the frame rate might change during the adaptation). Markers can be present as attributes within both the *gBSDUnit* and the *Parameter*. The use of markers within a gBSD enables the creation of application-specific gBSDs. More precisely, the markers contain semantically meaningful information that can be used by the transformation process. Note that some examples of gBSDs are also provided in Sect. IV.

C. Related Work

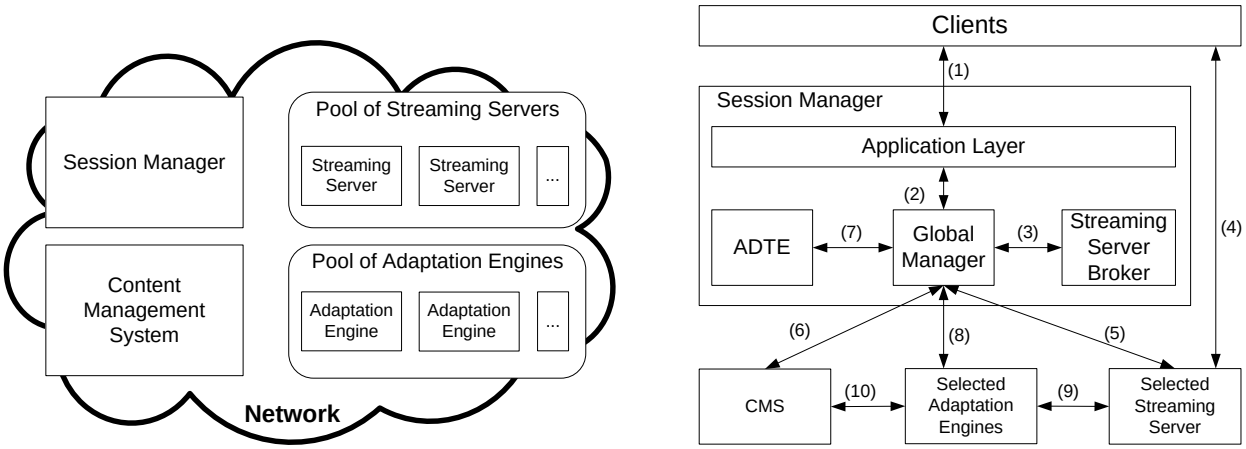
In recent years, a number of XML-driven content adaptation technologies have been developed, next to MPEG-21 gBS Schema.

The Bitstream Syntax Description Language (BSDL, [9]), which is part 5 of the MPEG-B standard, is built on top of the W3C XML Schema Language. This XML language allows to describe the structure of a (scalable) coding format in a so-called Bitstream Syntax Schema (BS Schema) [10]. BSDL comes with two standardized and format-agnostic parsers, which both take as input a BS Schema. The BintoBSD parser is responsible for producing XML descriptions, while the BSDtoBin parser is used for generating adapted bitstreams.

The Formal Language for Audio-Visual Object Representation, extended with XML features (XFlavor, [11]) is a declarative Java-like language. This tool provides means for describing the syntax of a bitstream on a bit-per-bit basis. It enables the automatic generation of a parser that is able to generate an XML description for a given bitstream. A tool to translate this XML description into an adapted bitstream is provided as well.

III. MuMiVA: Architecture

As discussed in the introduction, MuMiVA is a multimedia delivery platform that is able to transparently deliver multimedia content for heterogeneous usage environments.



(a) A global view on MuMiVA.

(b) Functioning of MuMiVA.

Fig. 2. Components of the MuMiVA platform in a fully distributed scenario.

Moreover, this transparent approach reflects in both a format-agnostic (i.e., independent of the underlying coding format) and application-agnostic characteristic (i.e., independent of the adaptations applied to the media resources). In this section, an overview is given of the MuMiVA architecture, together with its strengths and weaknesses.

A. Distributed Architecture: a Global View on MuMiVA

Our multimedia delivery platform contains multiple components that can be distributed across a managed network. Such a distributed architecture increases the scalability of the platform, since it allows extending the platform with additional components in order to anticipate an increasing load.

Fig. 2(a) gives an overview of the components present in our MuMiVA platform. In the current implementation of our platform, these components communicate by means of sockets. Explanatory notes for this figure are given below.

- *Content Management System (CMS)* is a multimedia archive that contains multimedia content encoded with (scalable) coding formats, as well as metadata about the content. Both structural (e.g., a gBSD of a particular bitstream) and semantic metadata (e.g., scene information for a particular video sequence) are present in the CMS.
- *Pool of Adaptation Engines* is a collection of distributed adaptation engines. An adaptation engine may deploy multiple adaptation techniques. In this paper, we discuss only one (format-agnostic) adaptation technique, i.e., gBS Schema. However, MuMiVA allows the adoption of other (format-agnostic or format-specific) adaptation tools. Examples of other adaptation tools are BSDL (format-agnostic) and transcoders (format-specific).
- *Pool of Streaming Servers* is a collection of distributed streaming servers.
- *Session Manager* couples the content servers, adaptation engines, and streaming servers. It provides a front-end for clients to the multimedia delivery platform by means of an application layer (e.g., webservice). Furthermore, this component selects the proper streaming server based on an assessment of the current server load. Finally, this component manages the different client sessions and is able to take decisions regarding the adaptation of the requested content, based on information regarding the usage environment of the clients.

Distributing the components of MuMiVA across the network makes it possible to achieve a scalable multimedia delivery platform. More specifically, a distributed multimedia system makes it easy to expand or contract its pool of servers to accommodate increasing or decreasing loads of the platform.

B. Functioning of MuMiVA

The MuMiVA architecture is shown in Fig. 2(b), as well as the communication between its different components. Explanatory notes for the chain of interactions between the different components of the MuMiVA platform are provided below.

(1) A client requests multimedia content by contacting the MuMiVA platform. Besides the requested content, the client also sends information about its usage environment.
(2) The application layer provides the information about the client to the global manager. The manager initializes a new session for each particular client. Consequently, the client receives a session ID.
(3) The global manager contacts the broker for the streaming server in order to select a proper streaming server based on the current server load. The selected streaming server will be announced to the client by the global manager through the application layer.
(4) The client connects to the selected streaming server, and provides its session ID in order to receive the desired content.
(5) The selected streaming server contacts the global manager, in order to receive information about the incoming request of the client, based on its session ID.

(6) The global manager fetches metadata about the requested content (e.g., bit rate and resolution), which are stored in the CMS.
(7) Once the global manager has all the necessary information at its disposal (i.e., metadata about the requested content and information about the usage environment), it contacts the Adaptation Decision Taking Engine (ADTE). The ADTE first decides which adaptation technique to use, based on the available adaptation engines. Once an adaptation technique is chosen (e.g., gBS Schema), the ADTE might take additional decisions related to the adaptation technique (e.g., the XML transformation tool in case of an XML-driven adaptation technique). Next, it calculates the adaptation parameters by matching the metadata about the requested content and the information about the usage environment (e.g., comparison of the resolution of a video sequence with the size of the screen of the end-user device). When the requested content includes both audio and video, the ADTE will select two adaptation engines: one for the video stream and one for the audio stream. In the latter case, the synchronization between the output of the adaptation engines is done by the streaming server.
(8) The global manager initializes the selected adaptation engines with the collected adaptation parameters.

Once the necessary negotiation between the client and MuMiVA is done, the client can start consuming the requested multimedia content. The streaming server, adaptation engines, and CMS form a pipeline system, where the communication is based on a pull-system. More specifically, the streaming server pulls content from the adaptation engine, which subsequently pulls content from the CMS. The following steps are performed during the streaming of the multimedia content to the client.

(9) The streaming server reads the adapted media resources from the adaptation engines. Subsequently, these adapted resources are send to the client by using the Real-time Transport Protocol (RTP, [12]). Real Time Streaming Protocol (RTSP, [13]) is used for the exchange of control operations between client and streaming server (e.g., the client can decide to pause the streaming of the content).
(10) The adaptation engines read the original media resources from the CMS. Furthermore, an adaptation engine customizes a given media resource according to the adaptation parameters received from the global manager. More detailed information about the internal working of an adaptation engine within MuMiVA is provided in Sect. III-C.

C. XML-driven Adaptation Engine

One the main features of our MuMiVA platform is the use of XML-driven content adaptation engines, based on MPEG-21 gBS Schema. As discussed in Sect. II, XML-driven content adaptation by using gBS Schema consists of three steps: gBSD generation, gBSD transformation, and bitstream generation. Within the MuMiVA platform, gBSD generation is seen as a preprocessing step. Consequently, the gBSDs of the corresponding media resources are already available in the CMS. Hence, the adaptation engines only need to perform the last two steps of the XML-driven content adaptation chain, i.e., gBSD transformation and bitstream generation.

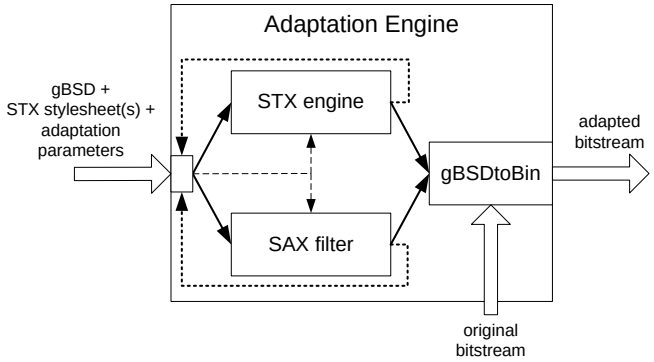


Fig. 3. Functioning of a MuMiVA adaptation engine, based on MPEG-21 gBS Schema. The dashed arrows denote the adaptation parameters and STX stylesheets. The bold arrows denote the data flow of the gBSD. The dotted arrows illustrate the possibility for multiple transformations of a gBSD during the adaptation.

Different XML transformation tools exist to transform a gBSD. There are two approaches to interpret and to transform XML documents. Firstly, traditional procedural programming languages such as Java or C++, together with a parser, can be used to consume XML data. Secondly, transformations can be implemented by using stylesheets together with a generic engine for interpreting these stylesheets (e.g., XSLT and STX). The main difference between the two approaches is the possibility to make use of generic software modules (transformation engines) in the latter case. Moreover, two main types of XML parsers exist. One built on top of tree-based models (e.g., Document Object Model (DOM)) and one on event-based models (e.g., Simple API for XML (SAX)). MuMiVA only uses SAX-based XML transformation tools, since these can be used in streaming environments (in contrast to DOM-based XML transformation tools) [14]. Therefore, two different transformation tools can be used within an adaptation engine of MuMiVA: a SAX filter (i.e., a Java program for XML transformations using a SAX-based parser) and a STX engine.

The functioning of an adaptation engine is depicted in Fig. 3. As discussed in the previous section, the adaptation engine receives adaptation parameters from the global manager. Furthermore, it reads the original bitstream from the CMS, together with its corresponding gBSD. When STX is used as XML transformation technology, the necessary STX stylesheets are also fetched from the CMS.

Based on the adaptation parameters, the adaptation engine selects the proper transformation tool (i.e., SAX filter or STX) and transforms the gBSD. Different transformations can be applied to the gBSD during the adaptation (see the dotted arrows in Fig. 3). For example, the first transformation may

consist of temporal rescaling by using a SAX filter; the second transformation may consist of scene extraction by using a STX engine. This will be further explained in Sect. IV.

After the transformation of the gBSD, the adapted content is generated by MPEG-21 DIA's gBSDtoBin parser, taking as input the transformed gBSD and the original bitstream.

It is important to notice that the adaptation engine is fully format-agnostic when STX is used, since STX uses a format-agnostic engine for executing a particular stylesheet. This is in contrast with the use of a SAX filter, which does not rely on format-agnostic logic.

D. Strengths of the MuMiVA Platform

In the previous section, we have discussed the overal architecture of our multimedia delivery platform, together with the internal functioning of its different components. Delivering multimedia content by using MuMiVA has the following benefits.

- *supports format-agnostic adaptation*: the use of XML-driven content adaptation implies that the adaptation engines can operate independent of the underlying coding format (i.e., the software modules are format-agnostic).
- *supports application-agnostic adaptation*: the use of XML-driven content adaptation also implies that the adaptation engines support application-agnostic adaptations, i.e., the software modules are independent of the application. In this context, an application corresponds to the kind of adaptation that is executed on the media resource (e.g., temporal scalability or scene selection).
- *extensible*: since our multimedia delivery platform is format-agnostic and application-agnostic, it can be considered straightforward to extend MuMiVA with new coding formats and applications.
- *scalable*: MuMiVA contains different components that can be distributed across the network, i.e., adaptation engines, streaming servers, and a CMS. It is rather straightforward to add or to remove these components from the MuMiVA platform. Hence, a high degree of scalability can be obtained.
- *support for varying usage environments*: when the usage environment of a client changes during the consumption of the requested multimedia content, the MuMiVA platform can dynamically change adaptation parameters according to the updated usage environment.
- *support for streaming*: the MuMiVA platform offers (adapted) multimedia content in a streaming environment. As discussed above, it implements the RTSP protocol, implying that mediaplayers such as VideoLan Client (VLC), Osmo, and QuickTime can play the content streamed by the MuMiVA platform. The adaptation of the content also occurs in a streaming fashion, since the XML-driven content adaptation engine is fully SAX-based [14].
- *interoperable*: it is important to make use of standardized and open technologies to obtain interoperability. The following technologies are used within the MuMiVA platform: MPEG-21 DIA, SAX, STX, RTSP, and RTP. Note that STX is not standardized yet; however, this technology is currently under consideration for standardization by W3C.
- *fully integrated*: to the best knowledge of the authors, the MuMiVA platform is the first multimedia content delivery platform that offers a fully integrated solution regarding format-agnostic and application-agnostic delivery of multimedia content in a streaming environment.

To illustrate the extensibility of our platform, we will discuss the necessary steps that need to be taken to extend the MuMiVA platform with the H.264/AVC Scalable Video Coding (SVC) format. It is important to notice that we did not yet provide support for SVC within MuMiVA due to the following reasons: the specification of SVC is not finalized yet, the RTP specification for SVC is still under development, and there is currently a lack of real-time SVC decoders. A straightforward application for SVC is the exploitation of scalability along its three scalability axes (i.e., temporal, spatial, and Signal-to-Noise Ratio (SNR)) [15]. The following steps need to be taken so that MuMiVA can provide support for the delivery of SVC bitstreams, adapted according to a certain usage environment.

- The SVC-compliant bitstreams, located in the CMS, need to be accompanied by their structural metadata (i.e., XML descriptions of the high-level structure of the bitstreams). These metadata will be used to adapt the bitstreams.
- One or more STX stylesheets or SAX filters need to be written, so that the adaptation engine is able to apply the necessary XML transformations to the XML descriptions. More specifically, these transformations correspond to the exploitation of different types of scalability in SVC bitstreams.
- Finally, the streaming server needs to be extended in order to enable the streaming of SVC-compliant bitstreams.

Note that, besides SAX filters, only new software has to be written for the streaming server. All other steps have no impact on the software of the MuMiVA platform. Consequently, the streaming server is not format-agnostic. However, MPEG-21 Digital Item Streaming (DIS, [16]), which is currently under development, provides a solution for format-agnostic streaming of multimedia content. In the future, MuMiVA could be extended with support for DIS, offering a fully format-agnostic solution for the adaptation and delivery of multimedia content.

IV. MuMiVA Applications

Two applications (i.e., video frame rate reduction and shot selection), which are deployed on our MuMiVA platform, are discussed in more detail in this section. Both applications are applied to two coding formats, i.e., MPEG-4 Visual [17] and H.264/AVC [18]. Since MuMiVA supports both coding formats, multimedia content compliant with these two coding formats is present in the CMS, together with its metadata (i.e., gBSDs describing the high-level structure of the bitstreams).

```
<gBSDUnit syntacticalLabel="bitstream" start="0">
  <!-- ... -->
  <gBSDUnit syntacticalLabel="RAU" start="0" marker="
      :shot=3|shot=4">
    <gBSDUnit syntacticalLabel="FRAME" start="0" length
        ="876" marker=":shot=4:fps=6"/>
    <gBSDUnit syntacticalLabel="FRAME" start="876"
        length="604" marker=":shot=4:fps=12"/>
    <gBSDUnit syntacticalLabel="FRAME" start="1480"
        length="597" marker=":shot=3:fps=24"/>
    <gBSDUnit syntacticalLabel="FRAME" start="2077"
        length="595" marker=":shot=4:fps=24"/>
  </gBSDUnit>
  <gBSDUnit syntacticalLabel="RAU" start="2672" marker=
      ":shot=4|shot=5">
    <gBSDUnit syntacticalLabel="FRAME" start="2672"
        length="945" marker=":shot=5:fps=6"/>
    <gBSDUnit syntacticalLabel="FRAME" start="3617"
        length="545" marker=":shot=4:fps=12"/>
    <gBSDUnit syntacticalLabel="FRAME" start="4162"
        length="675" marker=":shot=4:fps=24"/>
    <gBSDUnit syntacticalLabel="FRAME" start="4837"
        length="611" marker=":shot=5:fps=24"/>
  </gBSDUnit>
  <!-- ... -->
</gBSDUnit>
```

Fig. 4. Example of a gBSD for an H.264/AVC-encoded bitstream, containing information about the frame rate and the different shots.

Note that multiple gBSDs can be present for one resource. Indeed, as discussed in Sect. II-B, gBSDs can be application-specific due to the occurrence of markers. For simplicity, we assume that the gBSDs, present in the CMS, support both applications. More specifically, markers contain information regarding the frame rate and the different shots of the corresponding bitstream. An example of such a gBSD is given in Fig. 4. Note that the gBSD contains details up to frame level, which is sufficient for both applications.

A. Shot Selection

Shot selection enables the personalization of video content according to the preferences of a user. More specifically, a user can select individual shots out of a video sequence (e.g., goals in a soccer match). However, special attention needs to be payed to the extraction of the desired shots as the adapted bitstream needs to remain compliant with the corresponding specification. Therefore, we use the algorithm proposed in [19], where the gBSDs contain a description of the Random Access Units (RAUs). Each RAU contains a list of one or more frames, which in their turn belong to a particular shot. This is also illustrated in Fig. 4, which contains two RAUs (lines 3 and 9).

As depicted in Fig. 4, each *gBSDUnit* corresponding to a RAU contains a marker. This marker denotes, among other things, the shots that are located within the RAU. When a user selects a specific shot, only the RAUs containing frames which belong to the requested shot are kept during the transformation. Subsequently, within each selected RAU, frames which do not belong to the requested shot and which are located *after* the frames belonging to the requested shot (in decoding order), are dropped. Given the gBSD in Fig. 4, selecting shot 4 will keep only the two RAUs depicted in this figure. Furthermore, the last frame of the second RAU will be dropped, since it does not belong to shot 4. Note that the above discussed algorithm for shot selection can be applied to both H.264/AVC and MPEG-4 Visual.

```
<stx:transform pass-through="all" output-method="xml">
  <stx:param name="frame_rate" select="12"/>
  <stx:template match="gBSDUnit[@syntacticalLabel='
      FRAME']">
    <stx:if test="number(substring-after(@marker,':fps
        =')) &lt;= $frame_rate">
      <stx:process-self/>
    </stx:if>
    <!-- else: drop the gBSDUnit -->
  </stx:template>
</stx:transform>
```

Fig. 5. Simplified STX stylesheet for the exploitation of temporal scalability. *gBSDUnit*s are dropped based on the frame rate located in their marker.

B. Video Frame Rate Reduction

Reduction of the frame rate in a video sequence is obtained by dropping frames. This form of adaptation is also known as the exploitation of temporal scalability. Since the gBSDs already contain information regarding the frame rate, the transformation of the gBSDs can be considered straightforward for both H.264/AVC and MPEG-4 Visual. Indeed, as illustrated in Fig. 4, every *gBSDUnit* corresponding to a frame contains a frame rate. When the user requests a frame rate equal to 12 fps, each frame containing a higher frame rate than 12 is dropped. This is also illustrated by the simplified STX stylesheet that is depicted in Fig. 5.

C. Combining Shot Selection and Frame Rate Reduction

If the gBSDs support both shot selection and frame rate reduction, the two applications can be combined (i.e., selection of a particular shot, at a particular frame rate).

Two approaches are possible for this combination within the MuMiVA framework. A first option is the creation of a single STX stylesheet (or SAX filter) that can deal with both applications. This stylesheet would take two parameters as input: the requested shot and the frame rate. A second option is the development of two separate STX stylesheets (and/or SAX filters): one for each application. This is possible since an adaptation engine allows multiple XML transformations during the adaptation of media resources (as discussed in Sect. III-C). Note that the second approach provides more flexibility since it allows the reuse of the two transformation filters as stand-alone filters (e.g., when only shot selection is needed as an application).

V. Performance Results

In order to provide an estimate of the necessary computational power for our MuMiVA platform, we have evaluated the MuMiVA applications described in Sect. IV, i.e., exploitation of temporal scalability, shot selection, and the combination thereof, applied to MPEG-4 Visual and H.264/AVC. These applications were implemented by means of STX stylesheets. The gBSDs describe the bitstream syntax up to the level of a frame; however, in order to estimate the impact of the level of

TABLE I

BITSTREAM AND GBSD CHARACTERISTICS OF THE TEST SEQUENCES.

Format	Bitstream size (MB)	Frame rate (fps)	Length (s)	Bit rate (Mbit/s)	gBSD size (KB)	Coarse gBSD size (KB)
H.264/AVC	36.3	24	83.3	3.5	231	13
MPEG-4 Visual	72.1	24	83.3	6.9	392	n/a

detail of a gBSD, we have also created an additional gBSD for the H.264/AVC format which is detailed up to RAU level. Note that this coarse-grained gBSD is only suitable for a simplified version of the shot selection application (i.e., unnecessary frames are not removed inside the selected RAUs).

Characteristics of the used test sequences (having a resolution of 1280x512) can be found in Table I. Note that the gBSD for MPEG-4 Visual is more verbose than the gBSD for H.264/AVC due to the inclusion of parameters for each Video Object Sequence (VOS). The experiments were done on a PC having an Intel Pentium D 2.8 GHz CPU and 1 GB of system memory at its disposal. The operating system used was Windows XP Pro SP2, running Sun Microsystems's Java 2 Runtime Environment (Standard Edition version 1.5). The Joost STX engine (version 2006-10-5) was used, together with version 3.0.3 of MPEG-21 DIA's gBSDtoBin parser. The streaming server is based on the C++ library of Live555 Streaming Media. JProfiler version 4.2.1 and AQTime version 4 were used for profiling the adaptation engine and the streaming server respectively.

Table II tabulates the CPU usage of the adaptation engine and the streaming server (both executed on a separate core), together with the file sizes of the transformed gBSDs and the resulting bitstreams. A comparison between the adaptation engine and the streaming server in terms of CPU usage reveals that the adaptation engine operates very efficiently (from 3 % to 10 % average CPU usage), while the streaming server requires two times more CPU power than the adaptation engine (average CPU usage varying from 19 % to 25 %). The latter is due to the fact that the streaming server needs to parse the incoming bitstreams in order to correctly assign timestamps. This is in contrast with the adaptation engine, which can rely on the information located in the gBSD to perform the adaptations.

The following factors influence the CPU usage of the adaptation engine and/or the streaming server: application, coding format, and the level of detail of the gBSD (see Table II). It is important to remark that the STX engine takes the most CPU usage of the adaptation engine, since it includes the transformation logic (the gBSDtoBin parser mainly performs byte copy operations).

- *Application*: the CPU usage of the adaptation engine is application-dependent. Shot selection is more complex than temporal scalability because large parts of the sequence might be skipped (in case of shot selection), which implies that the transformation needs to be faster than real time during the skipping of particular shots. Combining two applications takes more CPU time; however, the second XML transformation (i.e., temporal scalability) only has to transform the gBSD fragments selected by the first transformation (i.e., shot selection). The CPU usage of the streaming server is application-independent as shown in Table II.
- *Coding format*: the CPU usage of both the adaptation engine and the streaming server is format-dependent. The adaptation engine takes more CPU usage for MPEG-4 Visual than for H.264/AVC due to the more verbose gBSD for the MPEG-4 Visual bitstream. This larger gBSD demands more CPU time during the XML transformation. The streaming server takes more CPU usage for H.264/AVC than for MPEG-4 Visual because the parsing process is more complex for H.264/AVC in order to assign proper timestamps to video frames.
- *The level of detail of the gBSD*: this factor only influences the CPU usage of the adaptation engine. This can be deduced from Table II by comparing the CPU usage of the shot selection application once for a gBSD with detail up to frame level and once for a gBSD with detail up to RAU level. The CPU usage of the adaptation engine increases when the level of detail of the gBSD increases. Obviously, the CPU usage of the streaming server is independent of the level of detail of the gBSD.

The memory usage is constant for both the adaptation engine and the streaming server (a maximum of 5 MB of memory is used). Moreover, applications, coding formats, and the level of detail of a gBSD have a negligible impact on the memory usage of these components.

TABLE II

FILE SIZES AND CPU USAGE FOR THE DIFFERENT MUMIVA SCENARIOS.

	H.264/AVC				MPEG-4 Visual		
	TS[a]	SS	SS+TS	SS_c	TS	SS	SS+TS
Transformed gBSD size (KB)	122	96	53	6	280	156	122
Adapted bitstream size (MB)	21.8	16.2	9.9	18.0	43.5	31.4	19.1
Adaptation Engine CPU_A[b] (%)	3	5	7	4	8	9	10
Adaptation Engine CPU_P (%)	5	8	10	6	15	20	25
Streaming Server CPU_A (%)	25	24	24	23	19	18	19
Streaming Server CPU_P (%)	50	50	50	50	45	45	40

[a]TS, SS, and SS_c denotes Temporal Scalability (obtain a frame rate of 12 fps), Shot Selection (select 25 shots out of 60 shots), and Shot Selection with a coarse-grained gBSD respectively.

[b]CPU_A and CPU_P denotes average and peak CPU usage respectively.

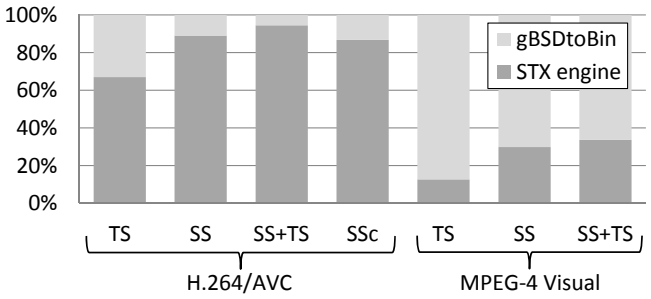


Fig. 6. Partition of the execution times (inclusive I/O operations) of the STX engine and the gBSDtoBin parser.

To examine the adaptation engine in more detail, the proportion in terms of execution times between the STX engine and the gBSDtoBin parser is depicted in Fig. 6. In case of the MPEG-4 Visual format, the gBSDtoBin parser takes most of the execution time (63 % to 83 %) because the bit rate of the MPEG-4 Visual sequence is twice as high than the bit rate of the H.264/AVC sequence. Hence, more I/O operations need to be performed by the gBSDtoBin parser for the MPEG-4 Visual bitstream. Furthermore, the application also influences the proportion between the STX engine and the gBSDtoBin parser, i.e., the STX engine will take more time when more complex transformations are executed.

VI. Future Extensions

In the near future, our MuMiVA platform will be extended with the following features.

- *Support for SVC*: MuMiVA will be extended with support for the SVC coding format (as discussed in Sect. III-D).
- *XML binarization*: the efficient representation of the XML descriptions is an important issue in the context of XML-driven content adaptation. It is important to use an XML binarization technology that allows efficient transformations of the XML descriptions in the binary domain in terms of execution speed.
- *Format-independent streaming*: as mentioned in Sect. III-D, providing support for format-independent streaming within the MuMiVA platform would allow to offer a fully format-agnostic solution for the adaptation and delivery of multimedia content. Note that the use of format-agnostic software modules implies the possibility to implement these modules in hardware, hereby obtaining hardware-accelerated, XML-driven content adaptation.
- *Other adaptation methods*: in the future, MuMiVA will be extended with additional adaptation methods (besides gBS Schema). Examples of other adaptation methods are MPEG-B BSDL and transcoders.

VII. Conclusions

In this paper, we have introduced MuMiVA, which is a multimedia delivery platform relying on XML-driven content adaptation engines. MuMiVA tackles the diversity in the current multimedia landscape by streaming multimedia content that is adapted according to the constraints of a certain usage environment. The multimedia content is customized using format-agnostic adaptation engines which, in their turn, use MPEG-21 gBS Schema as underlying technology.

An in-depth discussion of the architecture and functioning of our multimedia delivery platform has been provided. Furthermore, we have shown that MuMiVA is a fully integrated, extensible platform that supports varying usage environments and that is deployable in streaming environments. In order to demonstrate the flexibility of MuMiVA in terms of applications and coding formats, two different applications (i.e., exploitation of temporal scalability and shot selection) were applied to two different coding formats (i.e., MPEG-4 Visual and H.264/AVC).

Acknowledgment

The research activities as described in this paper were funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), the Belgian Federal Science Policy Office (BFSPO), and the European Union.

References

[1] A. Vetro, C. Christopoulos, and T. Ebrahimi, "Universal Multimedia Access," *IEEE Signal Processing Mag.*, vol. 20, no. 2, p. 16, March 2003.

[2] M. Amielh and S. Devillers, "Multimedia Content Adaptation with XML," in *8th International Conference on Multimedia Modeling (MMM'2001)*, Amsterdam, The Netherlands, November 2001, pp. 127–145.

[3] M. Kay, *XSLT Programmers's Reference, 2nd Edition*. Birmingham, UK: Wrox Press Ltd., 2001.

[4] P. Cimprich et al., "Streaming Transformations for XML," April 2007, Available on http://stx.sourceforge.net/documents/spec-stx-20070427.html.

[5] ISO/IEC, "21000-7:2004 Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation," October 2004.

[6] I. Burnett, F. Pereira, R. Van de Walle, and R. Koenen, Eds., *The MPEG-21 book*. John Wiley & Sons, March 2006.

[7] C. Timmerer, G. Panis, H. Kosch, J. Heuer, H. Hellwagner, and A. Hutter, "Coding Format Independent Multimedia Content Adaptation using XML," in *Proceedings of SPIE International Symposium ITCom 2003 on Internet Multimedia Management Systems IV*, vol. 5242, Orlando, September 2003, pp. 92–103.

[8] G. Panis, A. Hutter, J. Heuer, H. Hellwagner, H. Kosch, C. Timmerer, S. Devillers, and M. Amielh, "Bitstream Syntax Description: a Tool for Multimedia Resource Adaptation within MPEG-21," *Signal Processing: Image Communication*, vol. 18, no. 8, pp. 721–747, September 2003.

[9] ISO/IEC, "FDIS 23001-5:2007 Information technology – MPEG systems technologies – Part 5: Bitstream Syntax Description Language," January 2007.

[10] M. Amielh and S. Devillers, "Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content Adaptation," in *Proceedings of WWW2002: The Eleventh International World Wide Web Conference*, Honolulu, USA, May 2002, Available on http://wwwconf.ecs.soton.ac.uk/archive/00000185/01/index.html.

[11] D. Hong and A. Eleftheriadis, "XFlavor: Bridging Bits and Objects in Media Representation," in *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo (ICME)*, Lausanne, Switzerland, August 2002, pp. 773–776.

[12] RFC 3550, "RTP: A Transport Protocol for Real-Time Applications," Available on http://www.ietf.org/rfc/rfc3550.txt.

[13] RFC 2326, "Real Time Streaming Protocol," Available on http://www.ietf.org/rfc/rfc2326.txt.

[14] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, "Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments," *IEEE Trans. Multimedia*, vol. 7, no. 3, pp. 463–470, June 2005.

[15] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Extension of the H.264/MPEG-4 AVC Video Coding Standard," Accepted for publication in *IEEE Trans. Circuits Syst. Video Technol.*

[16] ISO/IEC, "FDIS 21000-18:2007 Information technology – Multimedia framework (MPEG-21) – Part 18: Digital Item Streaming," January 2007.

[17] ——, "14496-2:2004 Information technology – Coding of audio-visual objects – Part 2: Visual," May 2004.

[18] ITU-T and ISO/IEC JTC 1, "Advanced Video Coding for Generic Audiovisual Services," ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.

[19] S. De Bruyne, D. De Schrijver, W. De Neve, D. Van Deursen, and R. Van de Walle, "Enhanced Shot-Based Video Adaptation using MPEG-21 generic Bitstream Syntax Schema," in *Proceedings of the 1st IEEE Symposium on Computational Intelligence in Image and Signal Processing*, Honolulu, USA, April 2007, pp. 380–385.