PIECE-WISE MULTIMEDIA CONTENT ADAPTATION IN STREAMING AND CONSTRAINED ENVIRONMENTS[1]

Christian Timmerer[*], *Gabriel Panis*[†], *and Eric Delfosse*[‡]

[*]Klagenfurt University, Austria, christian.timmerer@itec.uni-klu.ac.at
[†]Siemens AG, Germany, gabriel.panis@siemens.com
[‡]IMEC, Belgium, eric.delfosse@imec.be

ABSTRACT

Universal Multimedia Access (UMA) has become a driving concept behind a significant amount of research activities. One of MPEG's (Moving Pictures Experts Group) responses to UMA is MPEG-21 Digital Item Adaptation (DIA). In this paper we present how tools as specified within DIA (i.e., normative XML-based description formats) are applied in streaming and constrained environments enabling piece-wise multimedia content adaptation including the adaptation decision-taking process and the actual resource adaptation in a coding format-independent way. Additionally, we demonstrate how the metadata overhead imposed by DIA tools can be reduced by means of appropriate metadata encoding tools.

1. INTRODUCTION

In order to cope with today's increasing diversity in multimedia content formats, networks and terminals, the development of an interoperable multimedia content adaptation framework has become a key issue. To address this issue, the Moving Picture Experts Group (MPEG) has specified a set of normative description tools within the MPEG-21 Digital Item Adaptation (DIA) standard [1][2] which can be used to support the adaptation. As such, DIA enables the construction of interoperable adaptation engines operating in a device- and coding format-independent way. Device independence is guaranteed through a unified description of the environment in which the content is consumed or through which it is accessed, namely the *Usage Environment Description (UED)*. Coding format independence is accomplished by the *Adaptation Quality of Service (AQoS)* and (*generic) Bitstream Syntax Description ((g)BSD)* tools. In this paper we will focus on AQoS and gBSD. AQoS enables the matching of bitstream characteristics to the usage environment context, described by the aforementioned UEDs, by providing content-related QoS information required to select the optimal adaptation parameters. These parameters together with the bitstream and corresponding gBSD are then the input of the actual bitstream adaptation engine. gBSD provides means for describing the structure of the bitstream independently from its actual encoding format, enabling a generic processor to generate a degraded version from the source bitstream by using its corresponding, possibly transformed, gBSD.

In practice, streaming scenarios require real-time adaptation according to dynamically changing usage environments within the multimedia content delivery chain [3]. Additionally, content-related QoS information such as peak signal-to-noise ratio (PSNR) and bit rate can be quite different from one bitstream segment to another. As a consequence, piece-wise multimedia content adaptation mechanisms – including the decision-taking process – are needed. Finally, mechanisms are required to limit the additional metadata overhead imposed by the aforementioned MPEG-21 DIA tools.

In this paper we present details of the adaptation approach based on [4] and extended by a piece-wise adaptation decision-taking process utilizing methods from [5]. This approach is explained on the basis of the simple use case where the bit rate of a video stream is adapted through selective B-frame dropping [6] in order to match the available network bandwidth. Finally, we demonstrate how applying appropriate metadata encoding schemes can reduce the metadata overhead.

The remainder of this paper is organized as follows. A brief overview of the adaptation engine is given in Section 2. Section 3 provides details about the adaptation decision-taking process on bitstream segments. The actual adaptation process of these segments is described in Section 4. Subsequently, Section 5 provides means for overcoming the verbosity imposed by XML-based

[1] Part of this work is supported by the European Commission in the context of the DANAE project (IST-1-507113); http://danae.rd.francetelecom.com. Special thanks to Ingo Wolf from T-Systems for providing additional test data (i.e., AQoS and gBSD for the audio content).

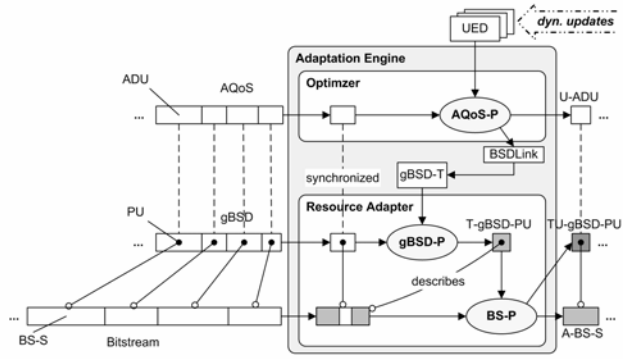

Figure 1 — Dynamic adaptation decision-taking and bitstream adaptation in streaming environments.

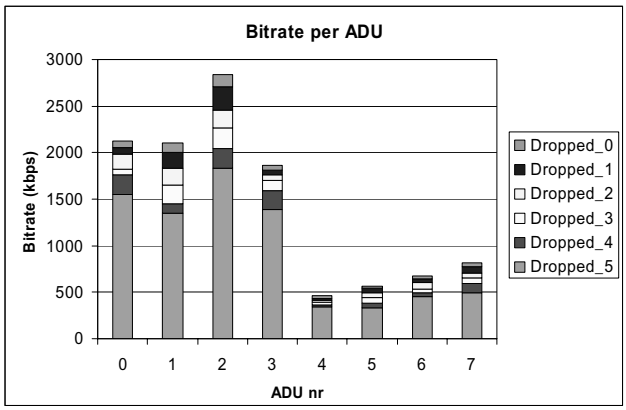

Figure 2 — Bit-rate per adaptation unit and per dropped number of B-VOPs.

metadata. The proposed approach is evaluated in Section 6 and Section 7 concludes this paper.

2. ADAPTATION ENGINE

The internals of the adaptation engine are depicted in Figure 1. AQoS information is provided on a per adaptation unit (ADU) basis, e.g., per scene or group of pictures (GOP). The actual bitstream is segmented according to the application requirements and described by a gBSD on a per process unit (PU) basis. According to [4], a PU comprises all information required for the adaptation of the bitstream segment described by this PU. The *AQoS processor (AQoS-P)* within the *optimizer* provides piece-wise adaptation decisions, according to the dynamically changing usage environment, which are used by the subsequent *gBSD processor (gBSD-P)* through the *BSDLink* description – another DIA tool. The gBSD-P produces a *transformed PU (T-gBSD-PU)* according to the decisions provided by the optimizer. The T-gBSD-PU is used by the *bitstream processor (BS-P)* which generates the *adapted bitstream segment (A-BS-S)* and is compliant to the gBSDtoBin process as specified within DIA. In order to enable further adaptation steps within the delivery chain the adaptation engine provides *updated gBSD (TU-gBSD-PU)* and *AQoS (U-ADU)* information.

3. ADAPTATION DECISION-TAKING

The adaptation decision-taking process is responsible for selecting the optimal adaptation parameters according to a given usage environment context on the basis of content-related QoS information. MPEG-21 DIA provides a tool, AdaptationQoS [1][5], to store and process this information.

Figure 2 depicts the bit rate evolution – per group of 10 frames (i.e., our ADU) – of an MPEG-4 visual elementary stream encoded compliant to the advanced simple profile at 25 frames per second and an average bit rate of 1,715 kbps. Furthermore, it presents the bit rate per dropped number of B-VOPs (Visual Object Planes) in an ADU (i.e., Dropped_x, with x number of B-VOPs). The choice of the actual B-VOP to be dropped is obtained according to [6].

As shown in Figure 2, the bit rate of a video stream can significantly differ from one bitstream segment to another. Thus, for optimal adaptation decision-taking it is desirable to provide QoS information per bitstream segment, i.e., per ADU. The size of an ADU can be chosen arbitrarily and is mostly triggered by trading of verbosity and accuracy. In this case an ADU corresponds to a sequence of 10 VOPs.

The AQoS tool allows for providing QoS information (cf. Figure 2) on an ADU per ADU basis through its switching mechanism as shown in Document 1. The `switchIOPinRefs` attribute indicates that the switching is done over the "ADUNR" parameter, which will be used by the AQoS-P to extract and process the QoS information of the corresponding ADU. When the AQoS-P receives a value for this parameter, it will match it to the value of the `switchValues` attribute, identifying the ADU, and select the corresponding `ContentDataSwitch` element. Each such element contains the bit rate information of the corresponding ADU. Hence, the AQoS-P will only process the bit rate information for the given ADU.

Finally, the presented switching mechanism enables the streaming and processing of each separate ADU in parallel with the bitstream and gBSD as shown in Figure 1. Indeed, the AQoS tool can easily be fragmented into its ADU constituents by extracting the corresponding `ContentDataSwitch` elements.

4. METADATA-DRIVEN CONTENT ADAPTATION

The output of the decision-taking process described above is a set of parameters which control the transformation of the gBSD. Generally, the transformation of the gBSD is

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS">
 <Description xsi:type="AdaptationQoSType">
  <Module xsi:type="LookUpTableSwitchType"
        switchIOPinRefs="ADUNR">
   <Axis iOPinRef="NR_B_FRAMES">
    <AxisValues xsi:type="IntegerVectorType">
     <Vector>0 1 2 3 4 5</Vector>
    </AxisValues>
   </Axis>
   <ContentSwitch iOPinRef="BITRATE">
    <ContentDataSwitch switchValues="0">
     <ContentValues xsi:type="FloatMatrixType"
         mpeg7:dim="5">
      <Matrix>
       2119.74   2052.24 1981.94
       1824.94 1760.50 1553.36
      </Matrix>
     </ContentValues>
    </ContentDataSwitch>
    <ContentDataSwitch switchValues="1">
     <ContentValues xsi:type="FloatMatrixType"
         mpeg7:dim="5">
      <Matrix>
       2103.68 1999.04 1836.86
       1646.68 1449.60 1348.56
      </Matrix>
     </ContentValues>
    </ContentDataSwitch>
    <!-- ... and so on ... -->
   </ContentSwitch>
  </Module>
  <IOPin id="ADUNR"/>
  <IOPin id="NR_B_VOPS"/><IOPin id="BITRATE"/>
 </Description>
</DIA>
```

Document 1 — AdaptationQoS description on a per adaptation unit (ADU) basis using the switching mechanism.

achieved through the application of an XSLT style sheet with the set of adaptation parameters produced by the AQoS-P as input parameters to the style sheet. In a streaming scenario, the gBSD-based adaptation concept and process [1] is applied at the level of gBSD fragments, called Process Units (PUs), instead of the entire gBSD. The fragmentation is arbitrary and application specific. The only requirement is that it should be possible to transform and process a PU individually without the need for information from another PU [3][4].

Figure 1 shows the adaptation engine architecture where the gBSD-based resource adaptation is applied using the PU fragmentation concept. The resource adapter is abstracted from the fragmentation process, which can be done either offline or online. In our current implementation the gBSD and bitstream fragmentation is done online because: a) the process is not resource intensive and b) it avoids storing potentially large number of files which implies multiple I/O accesses that are usually slow. In order to fully exploit the benefits of the PUs, the processing of the gBSD to extract the PUs can be done using the Simple API for XML (SAX[2]) interface, therefore reducing the runtime memory requirements.

The resource adapter enters a loop, requesting from the fragmentation algorithm to retrieve the next PU and the corresponding bitstream segment. The fragmentation algorithm steps through the gBSD and locates the next PU

[2] http://www.saxproject.org

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS">
 <Description xsi:type="BSDLinkType">
  <SteeringDescriptionRef uri="AQoS.xml"/>
  <BSDRef uri="gBSD.xml"
    processUnit="//dia:Description/
         gBSDUnit[@syntacticalLabel=':M4V:GOV']"
    maxPUSize="8262" maxBitstreamPUSize="50857"/>
  <BSDTransformationRef uri="transform.xslt"
    type="http://www.w3.org/1999/XSL/Transform"/>
  <Parameter xsi:type="IOPinRefType"
     name="nrBFrames">
   <Value>NR_B_VOPS</Value>
  </Parameter>
 </Description>
</DIA>
```

Document 2 — Example BSDLink description linking AQoS (cf. Document 1) and gBSD (cf. Document 3) and providing the PU context.

```
<dia:DIA
  xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
  xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS">
 <dia:DescriptionMetadata>
  <!-- -->
 </dia:DescriptionMetadata>
 <dia:Description xsi:type="gBSDType"
     addressUnit="byte" addressMode="Absolute"
     bs1:bitstreamURI="lotr.cmp">
  <gBSDUnit start="0" length="18"/>
  <gBSDUnit syntacticalLabel=":M4V:GOV"
      start="18" length="105947">
   <gBSDUnit syntacticalLabel=":M4V:I_VOP"
      start="18" length="12270"/>
   <gBSDUnit syntacticalLabel=":M4V:P_VOP"
      start="12288" length="7589"/>
   <gBSDUnit syntacticalLabel=":M4V:B_VOP"
      start="19877" length="3218"/>
   <!-- ... and so on; further VOPs ... -->
  </gBSDUnit><!--... further GOVs ...-->
 </dia:Description>
</dia:DIA>
```

Document 3 — gBSD fragment describing the bitstream at a frame and GOV level.

by trying to match the gBSD elements to the corresponding XPath[3] expression in the BSDLink description (Document 2). The BSDLink links the AQoS and gBSD and provides the information how to fragment the gBSD into PUs by means of an additional attribute defining the PU using XPath. Once the gBSD fragment is located and extracted, the addressing information in the gBSD and the corresponding bitstream segment (BS-S) is located in the resource and retrieved. Given that the PU and BS-S will be packetized into, e.g., Real-time Transport Protocol (RTP) packets for streaming, synchronization information is needed. The fragmentation algorithm uses hint tracks from the MP4 file format to identify the relationship between the current BS-S and corresponding Access Unit (AU). This relationship could be one-to-one, one-to-many or many-to-one.

The resource adapter receives: the PU, the corresponding bitstream segment, the timestamps for the PU, BS-S as well as the AU sequence number. The PU is then handled like an individual gBSD, as specified in [1]. This means the PU is transformed and then parsed to generate the adapted bitstream segment. In the case where

[3] http://www.w3.org/TR/xpath

the PU needs to be transported for possible adaptation somewhere else along the delivery path, an updated PU is also generated with aligned addressing information.

5. REDUCING THE VERBOSITY OF METADATA

The decision-taking and resource adaptation processes as described in the previous sections rely on XML-based metadata as specified within MPEG-21 DIA. Due to the metadata's plain-text nature it becomes a burden in constrained devices, especially relatively to the multimedia resource it describes which usually is available in a compressed and highly efficient representation format. As such, it becomes obvious to select an encoding scheme for plain text metadata as well, which provides (1) high compression ratios and (2) streaming capabilities for XML-based metadata. The MPEG-7 Systems Binary Format for Metadata (BiM) [7] satisfies both requirements and is therefore chosen for encoding the PUs and ADUs.

MPEG-7 BiM is an XML Schema-aware encoding scheme for XML documents, i.e., it uses information from the XML Schema to create an efficient alternative serialization of XML documents within the binary domain. One of the main features of BiM is that it enhances plain-text XML, which is non-streamable per se, with streaming capabilities. Therefore, BiM divides XML documents into access units (AUs) containing one or more fragment update units (FUUs). Each FUU includes the FU command (i.e., the decoder action – add, delete, update – for the corresponding fragment), the FU context (i.e., to uniquely determine the location of the fragment), and the FU payload (i.e., the actual XML data according to the context). In addition to the efficient binary representation and streaming capabilities, the FUU's bitstream structure offers the possibility to perform filter and update operation within the binary domain, i.e., similar to what is provided by scalable coding formats for multimedia resources. However, a special BiM configuration and AU/FUU organization is required which is not included in this paper due to space limitations.

6. EVALUATION AND MEASUREMENTS

We have evaluated our approach regarding the actual overhead imposed by the metadata. The results are shown in Table 1 with 10 VOPs per GOV, ADU, and PU respectively. It shows that plain text XML causes an overhead per GOV of about 3.86% which can be reduced through BiM encoding to 0.94% (for the visual stream). For audio we measured a larger overhead which is due to a fine-grained description of the gBSD and AQoS. Note that for BiM encoding the Zlib optimized codec for strings and the binary context path encoding was enabled.

Table 1 — Average GOV and textual/binary ADU and PU size and the resulting overhead with regard to the GOV size; compression ratio between plain text and BiM encoded ADU/PU.

Plain text	GOV size [bytes]	AQoS ADU size [bytes]	gBSD PU size [bytes]	Metadata Overhead [%]
visual	42,720.14	529.60	1,121.40	**3.86**
audio	1,957.50	813.00	9,160.00	**509.48**
BiM	**GOV size [bytes]**	**AQoS ADU size [bytes]**	**gBSD PU size [bytes]**	**Metadata Overhead [%]**
visual	42,720.14	279.00	121.00	**0.94**
audio	1,957.50	224.00	522.00	**38.11**
ratio (visual)		1.90	9.27	
ratio (audio)		3.63	17.55	

Additionally, the compression ratio between plain text and BiM encoded ADU/PU is given.

7. CONCLUSION

In this paper we presented further insights into the adaptation engine enabling interoperable multimedia content adaptation for UMA. Due to its generic and metadata-driven design this approach is applicable for streaming and constrained environments such as intermediary network nodes or mobile devices. The imposed metadata overhead can be reduced by appropriate encoding schemes although we see further improvements in this area which will be part of our future work.

8. REFERENCES

[1] ISO/IEC 21000-7:2004, "Information Technology — Multimedia Framework (MPEG-21) — Part 7: Digital Item Adaptation," 2004.

[2] A. Vetro and C. Timmerer, "Digital Item Adaptation: Overview of Standardization and Research Activities," to appear in *IEEE Trans. on Multimedia*, 2005.

[3] A. Hutter, P. Amon, G. Panis, E. Delfosse, M. Ransburg, and H. Hellwagner, "Automatic Adaptation of Streaming Multimedia Content in A Dynamic and Distributed Environment," submitted to the *IEEE Int'l Conference on Image Processing 2005*, Genova, Italy, Sept. 2005.

[4] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, "Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments," to appear in *IEEE Trans. on Multimedia*, 2005.

[5] D. Mukherjee, E. Delfosse, J.G. Kim and Y. Wang, "Optimal Adaptation Decision-Taking for Terminal and Network Quality of Service," to appear in *IEEE Trans. on Multimedia*, 2005.

[6] K. Leopold, H. Hellwagner, and M. Kropfberger, "QCTVA — Quality Controlled Temporal Video Adaptation," *Proceedings of the SPIE Int'l Symp. ITCom*, Orlando, USA, pp. 163-174, Sep. 2003.

[7] U. Niedermeier, et.al., "An MPEG-7 tool for compression and streaming of XML data," Proceedings of the 2002 IEEE Int'l Conf. on Multimedia and Expo (ICME), vol. 1, Lausanne, Switzerland, pp. 521–524, Aug. 2002.