Efficient Transformation of MPEG-21 Metadata for Codec-agnostic Adaptation in Real-time Streaming Scenarios[*]

Michael Ransburg, Hubert Gressl, Hermann Hellwagner
Klagenfurt University, Multimedia Communication Group
Universitätsstraße 65–67, Klagenfurt, Austria
{firstname.lastname}@itec.uni-klu.ac.at

Abstract

Scalable media contents, such as the new MPEG-4 Scalable Video Codec enable to easily retrieve different qualities of the media content by simply disregarding certain media segments. The MPEG-21-based codec-agnostic adaptation approach supports this concept by introducing an XML-based Bitstream Syntax Description (BSD) which describes the different segments of a media content. Based on this BSD, an adaptation node can intelligently adapt any scalable media (i.e., remove specific media segments) without the need for codec-specific knowledge. The adaptation approach consists of 1) transforming this BSD and 2) adapting the media based on the transformed BSD. In this paper, we focus on the BSD transformation step and evaluate different mechanisms w.r.t. their transformation efficiency given several application scenarios. In particular, we compare the traditional stylesheet-based mechanisms with a novel mechanism based on regular expressions. We discuss both mechanisms in terms of their expressiveness, and propose how to actually employ regular expressions for codec-agnostic adaptation. Finally, we quantitatively evaluate these mechanisms in different adaptation scenarios, which vary in the size and number of required BSD units.

1 Introduction

The last decade has brought about an impressive increase in the quantity of multimedia content available to an increasing number of different users (with different preferences) who access the content through a variety of devices and over diverse networks. In order to react to these heterogeneous usage environments, the designers of new media codecs attempt to include adaptation support into the codec design. These scalable media codecs support the generation of a degraded version of the original bitstream by means of simply removing bitstream segments. Depending on which segments are removed, the adapted version differs in one or more scalability dimensions, i.e., lower temporal or spatial resolution or quality. Note that not all of these dimensions are available for each codec type, i.e., video or audio.

The recently finalized MPEG-4 Scalable Video Codec (SVC) [9] is an example of a scalable codec. It segments the bitstream into Network Abstraction Layer Units (NALUs) where each NALU belongs to a specific temporal, spatial and quality layer in the media content, which is thereby organised in one base layer and several enhancement layers. Another example of a scalable media codec is the MPEG-4 Bit Slice Arithmetic Coding (BSAC) [6] audio codec, which separates each audio sample in one portion belonging to a base layer and up to 64 enhancement layer elements which can be truncated in order to retrieve a lower quality.

In Section 2 this paper introduces the MPEG-21-based approach to dynamic and distributed multimedia adaptation, which makes use of these scalable codecs. Further on, we focus on one particular requirement which results from this mechanism, i.e., to efficiently transform XML documents of various sizes. To this end, we introduce two well-known XML transformation mechanisms in Section 3 and compare them to our approach, which is based on regular expressions, in Section 4. Finally, we quantitatively evaluate and discuss the different approaches in Section 5 and we end with conclusions.

2 MPEG-21-based Dynamic and Distributed Adaptation

MPEG-21 Digital Item Adaptation (DIA) [8] supports scalable codecs by providing normative XML descriptions for 1) the current Usage Environment (UED), 2) the high-level Bitstream Syntax (BSD) of the scalable media content and 3) the available adaptation options (AQoS). The DIA-based adaptation approach consists of the following

---

[*] Work partially supported by the European project ENTHRONE II (IST-038463)

steps: 1) Based on the AQoS and the UED(s), an Adaptation Decision-Taking Engine (ADTE) decides which segments of the scalable media content to drop in order to meet the predefined QoS parameters of the session. 2) The BSD is transformed according to this adaptation decision, i.e., the description of certain bitstream segments is removed. 3) The bitstream is adapted according to the transformed BSD.

Since all of the DIA descriptions (including the transformation instructions for the BSD), are provided together with the media bitstream, this enables codec-agnostic adaptation nodes. These can support any type of scalable media which is properly described by such DIA descriptions.

The DIA-based adaptation mechanism was originally intended for static adaptation where the BSD describes the complete media bitstream and is transformed only once before the media bitstream is provided to the consumer. This is obviously inefficient in real-time streaming scenarios with dynamic usage environments. This static approach was therefore extended towards dynamic (and distributed) scenarios. It is enabled by a BSD which no longer describes the complete bitstream but only parts of it (e.g., access units or group of pictures). These BSD descriptions need to include timing information for their synchronized processing with the media content and are referred to as BSD Process Units (PUs). This fragmentation of the DIA-based adaptation mechanism also enables distributed adaptation (i.e., multiple adaptation steps along the delivery chain) where the BSD PUs are transmitted with the media segments which they describe. Consequently, the DIA-based adaptation process is performed on each of these BSD PUs (using up to date UEDs) which enables dynamic adaptation.

In contrast to static adaptation - where the BSD can have a considerable size of several megabytes - dynamic adaptation introduces further adaptation granularities. Depending on the application scenario, BSD PUs can describe NALUs, Access Units (AUs), or Group of Pictures (GoPs). The very fine NALU granularity has the advantage of a very low adaptation delay, since any adaptation decision is applied almost immediately. The coarser GoP granularity introduces adaptation delay, since adaptation decisions can only be applied at GoP boundaries. However, it is advantageous in distributed adaptation scenarios since higher compression factors can be achieved due to the larger sizes of the BSD PUs.

In any of the above cases, the BSD PUs need to be transformed according to the adaptation decision. Evaluating existing and our novel approach to this transformation process (in light of the different adaptation granularities introduced above) therefore represents the focus of this paper.

For further reading on MPEG-21-based dynamic and distributed adaptation, we refer to [5] and [7].

3 Existing XML Transformation Mechanisms

Extensible Stylesheet Language Transformations (XSLT) [4] is a declarative, template based, transformation language for XML documents. An XSLT processor needs two inputs: an XSLT style sheet that contains the transformation rules expressed in XML and the input XML document represented as DOM[1] tree. In addition, a set of parameters and parameter values can be passed to the XSLT processor to steer the transformation. The XSLT processor traverses the DOM tree and applies the changes according to the transformation rules defined in the style sheet.

Streaming Transformations for XML (STX) [3] uses style sheets with XSLT-like notation to perform the transformation of XML documents. Instead of a DOM representation of the XML document the event-based SAX[2] approach is used. Structural events are extracted from the input document and passed to the STX processor that filters or alters the events corresponding to the STX style sheet. In contrast to XSLT, the event-based STX approach does not mandate to have the complete document in memory, however this advantage comes at the cost of generating the SAX events. Additionally, this causes a lack of context information compared to DOM, therefore STX supports the buffering of events which makes it equally powerful as XSLT.

Both XSLT and STX support the codec-agnostic adaptation approach by providing a generic transformation process that is controlled by a codec-specific style sheet which can be provided together with the media content.

4 Regular Expressions for XML Transformation

Regular expressions [1] allow to specify a pattern for matching/replacing a substring in a string. They correspond to a type 3 grammar according to the Chomsky hierarchy [2] which creates a regular language recognizeable by a finite state automaton. Regular expressions are thus much less expressive than XSLT, which is Turing-complete and therefore represents a type 0 grammar. Similar to style sheets, regular expressions can be provided together with the multimedia content, only requiring a generic regular expressions processor at the adaptation node, thus supporting the codec-agnostic adaptation method.

In order to test their applicability to our application scenario of transforming BSD PUs, we implemented XSLT and STX style sheets for SVC and BSAC adaptation and then tried to realize the same functionality using regular expressions. We show an example BSD PU for SVC in Listing 1,

---

[1] Document Object Model, http://www.w3.org/DOM/

[2] Simple API for XML, http://www.saxproject.org/

which describes *start* and *length* of every NALU together with a *marker* which indicates priority, temporal id, spatial id and quality id, thus identifying which enhancement layer the described NALU belongs to. For SVC, the transformation involves disregarding *gBSDUnits* from the BSD (PUs) if the value of the *marker* indicates that the NALU belongs to a layer which shall be dropped according to the adaptation decision. For BSAC, the transformation additionally requires to update certain values in the BSD PUs.

For both cases, we were able to implement the corresponding regular expressions. Listing 2 shows the regular expression for SVC. This regular expression matches *gBSDUnits* with quality id between 1 and 9, i.e., removes quality enhancements layers 1 and 2 in our example. However, we encountered certain limitations which we describe below together with our approaches to encounter them.

Listing 1. BSD PU describing an SVC AU

```
<dia:DIA <!— namespaces omitted for brevity —>>
<dia:Description xsi:type=”gBSDType” addressUnit=”byte”
    addressMode=”Absolute”>
<gBSDUnit start=”0” length=”501” marker=”P0T0S0Q0”/>
<gBSDUnit start=”501” length=”815” marker=”P0T0S0Q1”/>
<gBSDUnit start=”1316” length=”1602” marker=”P0T0S0Q2”/>
<gBSDUnit start=”2918” length=”507” marker=”P0T0S1Q0”/>
<gBSDUnit start=”3425” length=”1605” marker=”P0T0S1Q1”/>
<gBSDUnit start=”5030” length=”3055” marker=”P0T0S1Q2”/>
<gBSDUnit start=”8085” length=”1375” marker=”P0T0S2Q0”/>
<gBSDUnit start=”9460” length=”5189” marker=”P0T0S2Q1”/>
</dia:Description></dia:DIA>
```

Listing 2. Regular Expression for SVC

```
<gBSDUnit(.*?)marker=\”P[0-9]T[0-9]S[0-9]Q[1-9]\”/>
```

Unlike style sheets, regular expressions are by itself not parameterizable, which is however needed to implement a certain adaptation decision provided by the ADTE. There are two solutions to this problem:

The obvious solution would be to extend regular expressions to be parameterizable, i.e., to introduce placeholders to the regular expressions which are replaced by the output from the ADTE. This replacement, i.e., the customization of the regular expression can again be performed by a regular expression. However, in order to enable this, additional control structures which steer this customization of the regular expression are necessary. These are traditionally provided by the programming language which uses the regular expressions and need to be defined, since they are not available in the generic regular expression processor in our application scenario.

A simpler solution is proposed which does not need any extensions to the normative regular expressions. The ADTE is a generic process which is steered by the AQoS. One possible layout for the AQoS is to contain tables which map a specific UED to an adaptation decision, e.g., the number of quality layers which shall be dropped from the media content in case that the available bandwidth drops to a certain value. We propose to align the AQoS description to include a regular expression (instead of the number of quality layers in the above example). This design change of the AQoS (which does not imply any changes to the DIA standard) allows to use the generic ADTE to provide regular expressions which steer the transformation of the BSD (PUs) in order to react to the given UED.

The BSAC requirement to update certain values in the BSD (PUs) leads to an additional requirement. That is, the regular expressions need to indicate whether they replace the matching substring by an empty string (i.e., disregard elements) or by another string (i.e., update values). For this we propose to adopt the corresponding Perl[3] syntax, i.e., `s/<regular expression>/<replacement string>/`.

To conclude, the design decisions described above allow the regular expressions to fullfill the same tasks as the style sheets for our application. Consequently, we perform a quantitative evaluation in the next section.

5 Evaluation and Discussion

In this section, we compare the time and memory needed for transforming SVC and BSAC (where applicable) BSD (PUs) of various sizes, i.e., at NALU, AU, GoP granularity (with GoP size 16), and additionally a BSD containing 3000 AUs in order to represent static, server-based adaptation. For STX, we use Joost[4], for XSLT we use libxslt[5] and for regular expressions we rely on the boost regular expressions library[6]. We only measure the time needed for the actual transformation and ignore any start up overhead (including, e.g., parsing the BSD and the style sheet), since their contribution to the overall CPU load is negligible in dynamic adaptation scenarios. We repeated all tests 500 times and only used the last 100 test runs for our measurements. This resulted in an insignificant deviation in the results, which is therefore not further considered. Additionally, for each test case, we measured the performance for disregarding *all* gBSDUnits (*dropall*), disregarding *no* gBSDUnit (*nothing*), and disregarding *half* of the gBSDUnits (*inbetween*) in order to cover different adaptation cases.

All tests were performed on a Dell Optiplex GX620 desktop with an Intel Pentium D 2.8 GHz processor and 1024 MB of RAM using Fedora Core 6 Linux with Kernel version 2.6.20 as an operating system. Memory consumption was measured using the *process status* (ps) tool and time measurements where performed based on the *gettimeofday* method.

Figures 1, 2, 3 and 4 show the results for transforming BSD (PUs) at NALU, AU, GoP and 3000 AU granularity, respec-

---

[3] Perl, http://www.perl.org

[4] Joost version 2007-07-18, http://joost.sourceforge.net

[5] libxslt version 1.1.21, http://xmlsoft.org/XSLT

[6] Boost.Regex version 1.33.1, http://www.boost.org/libs/regex/doc/

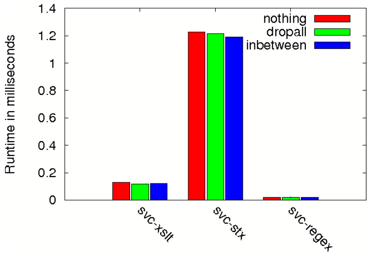

Figure 1. NALU granularity performance

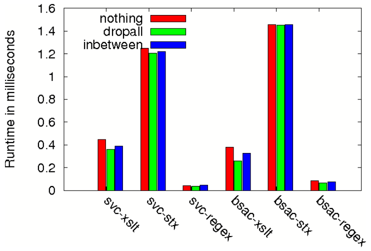

Figure 2. AU granularity performance

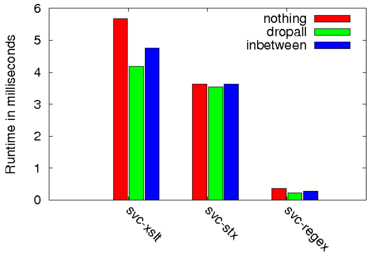

Figure 3. GoP granularity performance

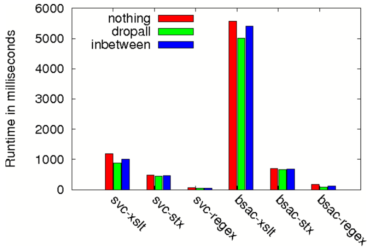

Figure 4. Performance for 3000 AUs

tively. As can be seen, regular expressions increase transformation performance at least by a factor of 4 compared to the other approaches. Since the transformation takes a considerable amount of time in our adaptation node [7] (on par with the actual adaptation of the bitstream), the usage of regular expressions significantly increases the throughput of the adaptation node. Additionally the measurements show that for small BSD PUs (i.e., at NALU and AU granularities) the XSLT mechanism performs significantly better than the STX mechanism. However, for larger BSD PUs STX is performing better than XSLT, which is particularly apparent for the BSD describing 3000 AUs. This is due to the event-based approach of STX which does not mandate to keep the complete XML document in memory. The break-even-point between STX and XSLT performance is at 60 kB / 74 AUs for BSAC and 4,8 kB / 8 AUs for SVC - apparently the additional update operations for BSAC are the reason for this difference. Memory consumption for small BSD PUs is insignificant, however for larger BSDs - such as our BSD example with 3000 AUs - memory consumption becomes significant for XSLT (i.e., 59.5 MB for SVC and 111.7 MB for BSAC) and slows down processing considerably as can be seen in Figure 4. Generally, regular expressions again perform best in terms of memory consumption.

6 Conclusion

In this paper, we described the state of the art in XML transformation in order to transform BSD (PUs) for MPEG-21-based codec-agnostic multimedia adaptation and introduced our novel approach, which relies on regular expressions. We qualitatively and quantitatively compared the different approaches. The results show that our approach is a viable alternative which significantly increases transformation performance, which in turn increases the throughput of our codec-agnostic adaptation node. Future work will analyse our approach's applicability to more complex adaptation scenarios.

References

[1] *IEEE Std 1003.1*. IEEE, 2004.

[2] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[3] P. Cimprich, O. Becker, C. Nentwich, H. Jirousek, M. Batsis, P. Brown, and M. Kay. Streaming Transformations for XML (STX) Version 1.0. Technical report, April 2007.

[4] J. Clark. XSL Transformations (XSLT) Version 1.0. Technical report, W3C, November 1999. W3C Recommendation.

[5] 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. In *Proc. ICIP*, Genova, Italy, September 2005.

[6] H. Purnhagen. An Overview of MPEG-4 Audio Version 2. In *Proc. International Conference on High-Quality Audio Coding*, Florence, Italy, September 1999.

[7] M. Ransburg, C. Timmerer, H. Hellwagner, and S. Devillers. Design and Evaluation of a Metadata-Driven Adaptation Node. In *Proc. WIAMIS*, Santorin, Greece, June 2007.

[8] A. Vetro and C. Timmerer. Digital Item Adaptation: Overview of Standardization and Research Activities. *IEEE Transactions on Multimedia*, 7(3):418–426, June 2005.

[9] T. Wiegand, J. Ohm, G. Sullivan, and A. Luthra. Special Issue on Scalable Video Coding - Standardization and Beyond. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9), September 2007.