TRANSFORMING MPEG-21 GENERIC BITSTREAM SYNTAX DESCRIPTIONS WITHIN THE BINARY DOMAIN

Christian Timmerer, Peter Lederer, and Harald Kosch

Department of Information Technology (ITEC), Klagenfurt University, Austria
{christian.timmerer, harald.kosch}@itec.uni-klu.ac.at, peter.lederer@edu.uni-klu.ac.at

Department of Information Technology (ITEC)
Klagenfurt University
Technical Report No. TR/ITEC/05/1.06
June 2005

TRANSFORMING MPEG-21 GENERIC BITSTREAM SYNTAX DESCRIPTIONS WITHIN THE BINARY DOMAIN

Christian Timmerer, Peter Lederer, and Harald Kosch

Klagenfurt University
Department of Information Technology (ITEC), Klagenfurt, Austria
{christian.timmerer, harald.kosch}@itec.uni-klu.ac.at, peter.lederer@edu.uni-klu.ac.at

ABSTRACT

XML-based metadata is widely adopted across the different communities and plenty of commercial and open source tools for processing and transforming are available on the market. However, all of these tools have the same requirement: they operate on plain text encoded metadata which may become a burden especially in constrained and streaming environments, e.g., when metadata needs to be processed together with multimedia content which is available in a highly efficient, binary representation format. In this paper we present techniques for transforming such kind of metadata which is encoded using the well known *MPEG-7 Systems Binary Format for Metadata (BiM)* without additional en-/decoding overheads, i.e., within the binary domain. As such it enables us to process both the multimedia data as well as the metadata within its compressed domain, e.g., for metadata-driven adaptation purposes within intermediary network nodes which are becoming increasingly popular in the multimedia community as well as in the XML community.

1. INTRODUCTION

More and more multimedia-enabled (mobile) devices are gaining access to advanced multimedia content through a plethora of access networks such as LAN, WLAN, GPRS, UMTS, etc. Research issues resulting from this are generally referred to as Universal Multimedia Access (UMA) [1], i.e., the content needs to be adapted according to the various terminal capabilities and network characteristics. Most recently, characteristics of users and her/his preferences have been taken into account as well during adaptation of the multimedia content [2][3]. Furthermore, this multimedia content is usually enriched with metadata providing support during the adaptation process [4] among others.

However, due to the fact that several parties are involved from the multimedia content production up to the consumption, interoperability during the delivery of the multimedia content is required which can be achieved by using standards provided by Moving Picture Experts Group (MPEG). The MPEG-21 Multimedia Framework – MPEG's most recent achievement – aims to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities [5]. Therefore, MPEG-21 introduces the concept of Users interacting with Digital Items. A User is defined as any entity (including individuals, communities, organizations as well as software agents) that interacts in the MPEG-21 environment or makes use of Digital Items. A Digital Item is referred to as a structured digital object with a standard representation, identification and metadata, i.e., a container for different kinds of resources and metadata represented within a standardized XML-based structure.

A vital part of MPEG and important with regard to UMA is part 7 entitled Digital Item Adaptation (DIA) [6]. DIA provides – among others – normative description tools enabling the construction of device and coding-format independent adaptation engines. An integral part of DIA is the description of the multimedia content's bitstream syntax, i.e., how it is organized in terms of frames, layers or packets, in a generic way. The resulting XML-based metadata document is referred to as generic Bitstream Syntax Description (gBSD). This gBSD is transformed according to the constraints imposed by the usage environment using standardized and open source XML transformation tools such as the Extensible Stylesheet Language Transformation (XSLT) or the Streaming Transformation for XML (STX). Such transformation usually includes removing and updating portions of the bitstream. Subsequently, the transformed gBSD is processed by a generic adaptation module which copies only the remaining portions (according to the transformed gBSD) from the source bitstream to the target bitstream. As such, only one adaptation module is required for adapting bitstreams encoded in different scalable coding formats described by gBSD. The general architecture for this gBSD-based adaptation approach is depicted in Figure 1.

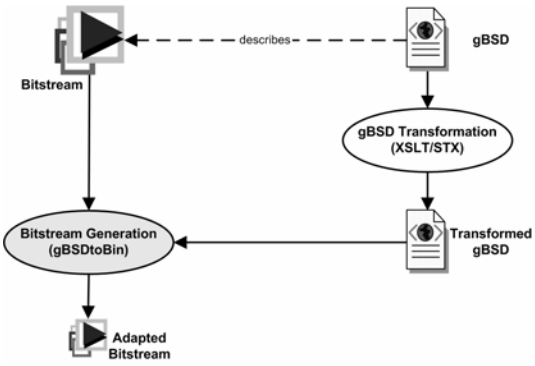

Figure 1 — gBSD-based Adaptation Architecture.

In practice however, the gBSD transformation is performed on plain text XML descriptions which imposes some burdens as we will see in the following section. In this paper, however, we present a novel approach for transforming gBSDs within the binary domain, thus, reducing the metadata overhead and improving the overall performance of the adaptation process.

The remainder of this paper is organized as follows. Section 2 provides an overview of the background and motivation. In Section 3 a novel approach for transforming such gBSDs within the binary domain is introduced. Sections 4 and 5 give detailed information how we evaluated our approach with a subsequent discussion of the results in Section 6. Related work will be discussed in Section 7. The Paper will be concluded in Section 8.

2. MOTIVATION AND BACKGROUND

In recent years research regarding multimedia content adaptation is heading towards adapting the content within the delivery chain, i.e., at certain selected network nodes, instead of traditional server-side adaptation [7][8]. This is required due to the fact that it is not conceivable that one single adaptation module can be deployed capable of reacting to all kind of usage environments. It is rather expected that more and more independent multimedia adaptation services will emerge. However, in order to exploit such services, metadata needs to be delivered alongside with the content.

Multimedia content is usually encoded using highly efficient encoding techniques and its processing can be achieved in a similar way. On the other hand, metadata is mainly XML-based and encoded in plain text and when transporting together with the content, bandwidth requirements are increasing which conflict with the aforementioned achievements regarding efficiency. Additionally, new trends in multimedia content adaptation focus on the adaptation in the compressed domain by exploiting scalable coding formats such as MPEG-4 and JPEG2000. New emerging standards like MPEG-4 Scalable Video Coding (SVC)1 confirm this trend.

Thus, the metadata overhead needs to be reduced by means of appropriate encoding schemes. Traditional compression techniques are inadequate as they ignore the XML structure and no streaming support is given. Our choice fall therefore on the MPEG-7 Binary Format for Metadata (BiM) [9] which allow random access to selected XML-subtrees and furthermore defines useful operations on these sub-trees. Consequently, we will parse and manipulate the binary encoded metadata within the binary domain.

In this paper we present techniques for filtering and updating such metadata fragments within the binary domain by means of MPEG-7 BiM. Therefore, special configuration settings during the metadata encoding process are required which are the main focus of this paper. In the sequel and for the sake of completeness we give a brief overview of BiM.

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. This schema knowledge enables the removal of structural redundancy, e.g., element and attribute names, which achieves high compression ratios with respect to the document structure. Furthermore, element and attribute names as well as data are encoded using dedicated codecs based on the data type (integer, float, string) which further increases the compression ratio. However, one of the main features of BiM is that it provides streaming capabilities for XML-based data which is one of the main disadvantages of plain text XML. Therefore, BiM divides the XML tree into access units (AUs) containing one or more fragment update units (FUUs). Each FUU includes the FU *command*, FU *context*, and FU *payload* which are described briefly in the following:

- The *command* specifies the decoder action for the corresponding fragment which can be either add, delete, replace, or reset, i.e., BiM also provides partial updates of an XML document.
- The *context* is used to uniquely determine the location of the fragment in the XML document.
- The *payload* contains the actual XML data according to the context.

Figure 2 illustrates how an XML document is divided into AUs and streamed over the network. In particular, it shows how a sub-tree of the whole XML document is transmitted over the network and added to the description tree at the receiver side (cf. dotted line).

By definition, each AU can be decoded separately while ensuring validity against the corresponding XML

[1] see http://www.chiariglione.org/mpeg/ for details

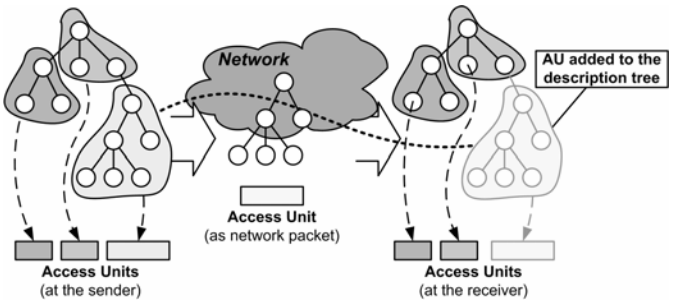

Figure 2 — Streaming XML Documents over the Network by using Access Units.

Schema. The FUUs are processed according to the FU command, i.e., *added* to, *deleted*, or *replaced* from the (partially) instantiated XML document. The *reset* command resets the BiM decoder and starts again with the initial description tree. Especially the replace command enables selective updates of (parts of) a document.

Finally, the FUU specification allows to perform filter operations within the binary domain, i.e., by means of simple bit pattern matching instead of time-consuming string comparisons as we will show in the next section.

3. BINARY TRANSFORMATION OF MPEG-21 METADATA

3.1. Usage scenarios

Our work is motivated by two usage scenarios which may benefit from the gBSD-based adaptation as introduced in Section 1:

(A) Adapting streaming audio/video (AV) resources according to network bandwidth constraints and user preferences.

(B) Adapting images according to terminal capabilities.

For scenario (A) we have considered MPEG-4 Advanced Simple Profile (ASP) Visual Elementary Stream (VES) which has been described with a gBSD at two different levels of detail: at a visual object plane (VOP) level enabling *B-VOP dropping* (also known as B-frame dropping) and at a scene level which allows personalization of the AV resources by means of, e.g., violent *scene dropping*. The fragment of such a gBSD is shown in Document 1. Each VOP is described by a corresponding `gBSDUnit` element and the VOP type is indicated within the `syntacticalLabel` attribute. Scenes are described by introducing an additional layer of `gBSDUnit` elements comprising an arbitrary number of child `gBSDUnit` elements belonging to the same scene. The type of scene is characterized by means of the `marker` attribute and its actual value depends on the application requirements, e.g., describing the level of the violence for each scene.

```
<dia:DIA><dia:Description xsi:type="gBSDType"
  addressUnit="byte" addressMode="Absolute"
  bs1:bitstreamURI="content/bsone.cmp">
  <gBSDUnit syntacticalLabel=":M4V:VO" start="0"
      length="4"/>
  <gBSDUnit syntacticalLabel=":M4V:VOL"
      start="4" length="14"/>
  <gBSDUnit start="18" length="520176"
      marker="ICRAParentalRatingViolenceCS-2">
    <gBSDUnit syntacticalLabel=":M4V:I_VOP"
        start="18" length="13522"/>
    <gBSDUnit syntacticalLabel=":M4V:P_VOP"
        start="13540" length="15128"/>
    <gBSDUnit syntacticalLabel=":M4V:B_VOP"
        start="28668" length="2734"/>
    <gBSDUnit syntacticalLabel=":M4V:B_VOP"
        start="31402" length="2714"/>
    <!--...  and so on, i.e., further VOPs ...-->
  </gBSDUnit>
  <!--... and so on, i.e., further scenes ...-->
</dia:Description></dia:DIA>
```

Document 1 — Fragment of a gBSD describing an MPEG-4 ASP VES at VOP and scene level.

For scenario (B) JPEG2000 encoded images have been used which provide 3 levels of scalability, namely *spatial*, *quality*, and *color reduction*. The fragment of a gBSD describing a JPEG2000 image is shown in Document 2. This fragment describes the relevant parts within the main header of JPEG 2000 images that need to be updated in case of color reduction is applied. The `:J2K:Csiz` parameter contains the number of color components and, for example, a value of 1 would result in a grayscale image and the subsequent `gBSDUnit` elements which are marked with `C0`, `C1`, and `C2` have to be removed accordingly.

In the next section we describe how the metadata, i.e., the gBSD, can be mapped to MPEG-7 BiM access units (AUs) and fragment update units (FUUs) in enabling these transformations within the binary domain.

3.2. Mapping metadata to BiM access units

In order to perform a binary transformation the plain text encoded gBSD has to be mapped to MPEG-7 BiM AUs and FUUs using a special encoder configuration and has to be BiM encoded afterwards.

The encoder configuration is derived from the syntactic and semantic information of the gBSD and based on the required transformation operations. The semantic information is provided through the `marker` attribute whereas the syntactic information can be found in the `syntacticalLabel` attribute. Additionally, the structure of the gBSD, i.e., the position, hierarchy and layout of the gBSD elements, is exploited during the transformation process as described in Section 3.3. The required transformation operations result from the nature of scalable multimedia formats and can be categorized in remove and (minor) update operations. Removing of gBSD elements refers to the removal of corresponding bitstream segments such as temporal or spatial

```
<dia:DIA><dia:Description xsi:type="gBSDType"
  addressUnit="byte" addressMode="Absolute"
  bs1:bitstreamURI="content/city.jp2">
 <gBSDUnit syntacticalLabel=":J2K:MainHeader"
    start="0" length="135">
  <gBSDUnit syntacticalLabel=":J2K:SIZ" start="2"
     length="49">
   <gBSDUnit addressMode="Consecutive"
      length="49">
    <!-- other parameters ... -->
    <Parameter name=":J2K:Csiz" length="2">
     <Value xsi:type="xs:unsignedShort">3</Value>
    </Parameter>
    <gBSDUnit syntacticalLabel=":J2K:Comp_siz"
      length="3" marker="C0"/>
    <gBSDUnit syntacticalLabel=":J2K:Comp_siz"
      length="3" marker="C1"/>
    <gBSDUnit syntacticalLabel=":J2K:Comp_siz"
      length="3" marker="C2"/>
   </gBSDUnit>
  </gBSDUnit>
  <!-- further parameters ... -->
 </gBSDUnit>
 <!-- :J2K:Tile with :J2K:Packets + :J2K:EOC -->
</dia:Description></dia:DIA>
```

Document 2 — Fragment of a gBSD describing an JPEG2000 image.

enhancements layers of videos. Updating of gBSD elements refers to the update of corresponding bitstream parameters such as updating the number of remaining enhancement layers. Therefore, updating is mostly required following a remove operation.

In both cases, remove and update, the concerned FUU is identified first and the actual operation is applied subsequently. Note that in some cases the operation is not only applied on the actual element but also on one or more following FUUs. In the following we present how gBSDs for our two scenarios needs to be mapped to AUs and FUUs in order to enable these operations within the binary domain.

For scenario (A) the gBSD is encoded as follows: `gBSDUnit` elements describing VOPs (I-, P- or B-VOPs) are represented by two consecutive FUUs. The first FUU comprises only the `syntacticalLabel` attribute whereas the actual content (i.e., element name and remaining attributes) is encoded in the second FUU. Furthermore, `gBSDUnit` elements representing scenes are divided into three consecutive FUUs. The first FUU contains only the `marker` attribute which is required for identification of the corresponding scene. The remaining FUUs comprise the `start` and `length` attributes respectively which are possibly updated during the transformation process. Finally, all these FUUs are packed to an open number of AUs depending on the application requirements. The only restriction is that the two groups of FUUs mentioned above must not be divided to separate AUs.

Conclusion. In order to remove gBSD elements within the binary domain the attribute value which is used for making such a decision needs to be encoded into one

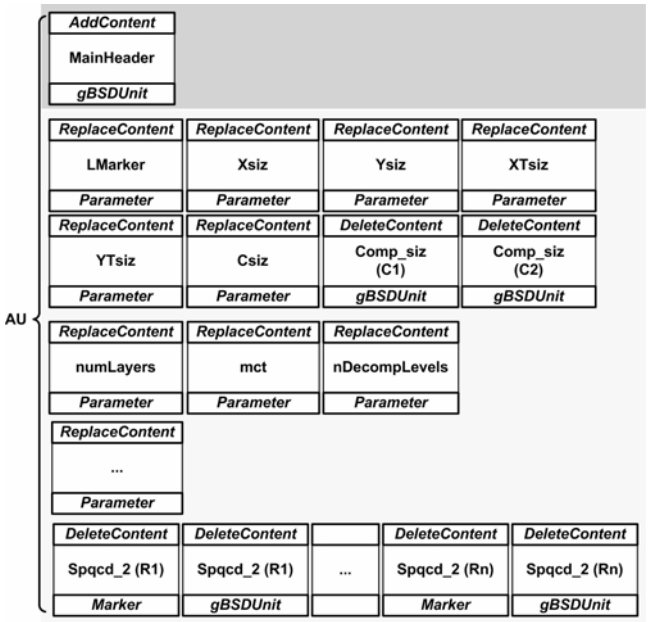

Figure 3 — Mapping of a gBSD describing the main header of an JPEG2000 image to a BiM AUs and FUUs.

FUU. Subsequent FUUs are removed – if necessary – if they have the same context path or a subset thereof.

For scenario (B) the configuration of the BiM encoding is more complex due to interdependent remove and update operations. For instance, color reduction removes `gBSDUnit` elements which are no longer required (i.e., packets marked with a predefined pattern) and updates certain `Parameter` elements (e.g., the `:J2K:Csiz` parameter from Document 2). In general, the gBSD is divided into three AUs. The first one contains information about the main header of the image, the second one describes the tile, and the last one represents the end of codestream (EOC) flag.

The AU which represents the main header can be divided into two major parts. The first part includes one single FUU comprising the `gBSDUnit` element describing the complete main header. The second part consists of an arbitrary number of FUUs representing different attributes and parameters of the main header which probably have to be updated or deleted during the transformation process.

The second AU, i.e., the tile, can be divided into three consecutive logical blocks. The first block consists of a FUU which includes all the information about the tile header and three additional FUUs which encode the attributes `syntacticalLabel`, `start`, and `length` of the tile. The second logical block contains only one single FUU representing the `:J2K:Psot` parameter of the tile header which indicates the length of the tile. This parameter always has to be updated during the transformation process. The last block contains a list of packets and each packet is mapped to four consecutive FUUs. These FUUs encode the `marker` and `length`

attributes of a Packet, the actual content, and the value of the `:J2K:Nsop` parameter. This `:J2K:Nsop` parameter represents an ascending sequence number of the packets within a tile which needs to be updated if packets are removed.

The mapping of a gBSD describing an JPEG2000 image (Document 2) to BiM AUs and FUUs is excerpted in Figure 3. Each box in the figure represents one single FUU and the upper sub-box contains the FUCommand. The syntactical name of the gBSD is provided in the middle of the box, and the bottom sub-box identifies the corresponding gBSD element type, i.e., `gBSDUnit` or `Parameter` elements or `marker` attribute.

Conclusion. In order to update gBSD element or attribute values within the binary domain the actual value needs to be encoded into one single FUU which is simply replaced by a pre-encoded bit pattern during the transformation process. The FUU is identified using the context path only.

3.3. Transformation within the binary domain

Our approach is based on three principles. First, the gBSD is encoded, i.e., fragmented, using MPEG-7 BiM as described in Section 3.2. Second, a binary filter module accomplishes fast and easy access to the AUs and FUUs respectively. Depending on the result of the filtering process a fragment can be either *added to* or *deleted from* the BiM bitstream *without decoding* its payload. And third, FUUs representing `Parameter` elements or attributes can be updated directly within the binary domain without decoding them. In order to achieve such an update, the binary filter module is used to identify the corresponding fragments by means of the context path. Subsequently, the payload of the fragment can be simply replaced by a new binary pattern which is created using the BiM payload encoding algorithm.

The binary filter module takes advantage of the structure of the binary context path of an encoded FUU which generally consists of a list of node names followed by a list of corresponding position codes as depicted in Figure 4. Thus, this kind of bitstream organization facilitates two different filtering modes: test the structure of a fragment and find the unique position of a fragment within the whole document, i.e., the gBSD. In the former case only part one of the path, i.e., the list of node names, is compared with a certain binary filter expression. This is used for identifying elements being possibly removed from the gBSD based on the content of the payload which contains only the value of one single attribute. For example, a filter expression could check if a fragment contains one of the `gBSDUnit` elements which – in the plain text gBSD – could be identified by the XPath expression `//dia:DIA/dia:Description/gBSDUnit`. This feature is used in the transformation

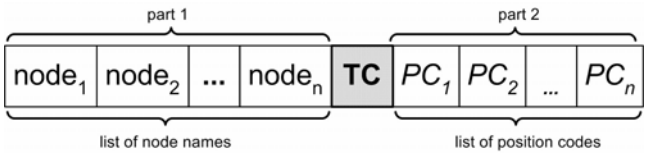

Figure 4 — Structure of binary context path encoding.

method of B-VOP dropping where all `gBSDUnit` elements describing B-VOPs are removed from the binary encoded gBSD. In the latter case (i.e., find the unique position of a fragment within the whole document) both parts of the binary context path are compared with a filter expression to determine the unique position of a fragment within the document. As such, a single gBSD element could be easily identified, e.g., to find certain `Parameter` elements which have to be updated as heavily used for scenario (B).

Additionally, the context path has one more convenient feature. The path of a parent node is a subset of the path to all child nodes of this parent. Thus, it could be easily used to apply the same transformations (e.g., add or remove) to all child nodes. This property is used, e.g., in combination with the transformation method of "scene dropping" to identify all frames which belong to a certain scene and immediately delete them from the bitstream.

3.4. Binary Transformation Framework (BTF)

For evaluation purposes we have implemented the functionality described in the previous section within a Binary Transformation Framework (BTF) which basically consists of the following parts:

- **BTF encoder**: this module maps the textual gBSD to a binary stream. It includes a MPEG-7 BiM encoding module which is steered by a special configuration file. This file contains the "rules" how to fragment the XML document and how to create the corresponding AUs and FUUs.
- **BTF transformer**: this transformation module is the main part of our framework. Using a number of special application- and/or user-defined parameters, it performs a binary gBSD transformation directly on the bitstream created by the BTF encoder. These parameters include some structural information about the bitstream as well as the transformation method (e.g. scene dropping for video gBSDs) and some user specific choices (e.g. which scene to remove).
- **BTF decoder (optional)**: this module decodes the bitstream which contains the fragmented gBSD using an MPEG-7 BiM decoding module and re-creates one single XML file from all the decoded fragments.

Figure 5 illustrates the high-level architecture of this framework. Our framework implements two different transformation methods for video descriptions and three

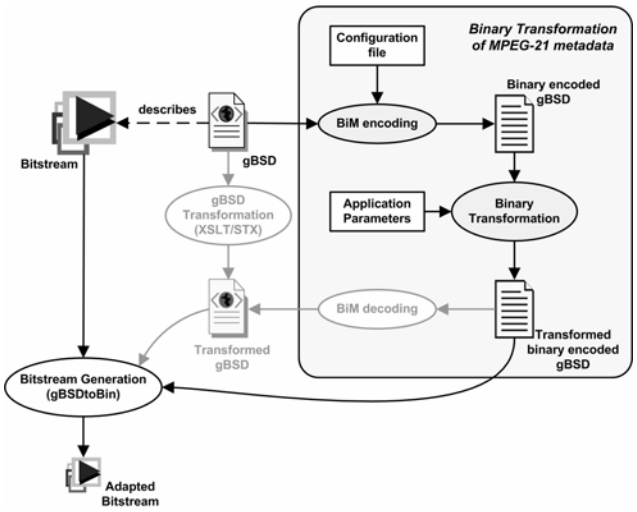

Figure 5 — High-level architecture for binary transformation of MPEG-21 metadata.

different transformation methods for image descriptions as described in Section 3.

4. EXPERIMENTAL SETUP

In order to measure the performance of the binary transformation of MPEG-21 metadata using our BTF, we have conducted a number of experiments which apply different transformations on MPEG-21 gBSD files.

4.1. Datasets

As input data we have used two types of gBSDs describing MPEG-4 Visual Elementary Streams (VESs) compliant to the Advanced Simple Profile (i.e., FOREMAN, AKIYO) and describing JPEG2000 images (i.e., CITY, SHANGHAI). All test sequences have been taken from the MPEG-21 test data set.

The gBSDs describing the MPEG-4 VESs have been transformed enabling bit rate reduction of the actual media bitstream by means of B-VOP dropping. Personalization of the content is achieved through scene dropping. For gBSDs describing JPEG2000 images three different transformation modes have been applied: color, quality, and spatial reduction.

4.2. Testing modes

We have performed our test in three different modes:

(TM-1) Transformation of plain text encoded gBSDs using regular XSLT style sheets and a legacy XSLT processor.

(TM-2) gBSDs are BiM encoded but the transformation is still performed on plain text encoded gBSDs as described in (TM-1). As such, the BiM encoded gBSD has to be decoded before being transformed and re-encoded afterwards.

(TM-3) Transformation of BiM encoded gBSDs within the binary stream as described in Section 3.

(TM-1) refers to the use case where only XSLT capabilities are available at the transformation device. However, this implies that the gBSD is transmitted in plain text over the network in the case this device is located somewhere in the network which increases bandwidth requirements of the overall multimedia delivery session. The second mode, i.e., (TM-2), assumes that BiM en-/decoding capabilities are available which drastically reduces the bandwidth requirements but increases processing efforts due to the de-/encoding overhead before and after the actual XML transformation by means of legacy XSLT processors. Finally, in (TM-3) the transformation device is equipped with the binary transformation framework as proposed in this paper.

4.3. Test environment

Our test environment consists of a small JAVA application which is able to

- load, process, and store XML files using the JDOM[2] API,
- apply XML transformations using JAVA XSLT processing methods of the JDOM API,
- start and use methods (en-/decode) of the MPEG-7 BiM reference software, and
- start and use methods (transform) of the BTF.

All experiments have been performed a 2.5GHz desktop computer equipped with 512MB RAM. The test machine is using the Microsoft Windows XP operating system.

5. RESULTS

The results of our experiments are shown in Table 1 and Figure 6 respectively. Table 1 provides the runtime measurements of the different transformations and test modes. The first two columns contain information about the test data and transformation method used, i.e., CITY and SHANGHAI describe JPEG2000 images whereas FOREMAN and AKIYO describe MPEG-4 VESs. The next three columns present the actual results of the experiments in milliseconds as an average value of five consecutive measurements. The third column represents the results for (TM-1), the fourth column the results for (TM-2), and the last column contains the results for (TM-3). Figure 6 provides average runtime measurements of the three test modes among the test data, i.e., the average for each test data has been calculated using the values from Table 1.

[2] www.jdom.org

Table 1 — Runtime results of the different transformations and test modes.

Test data	Transform. Method	(TM-1) [ms]	(TM-2) [ms]	(TM-3) [ms]
CITY	color	1692.60	3555.27	402.00
	quality	1630.60	3493.27	360.00
	spatial	1898.80	3761.47	526.00
SHANGHAI	color	2930.00	6989.83	2508.00
	quality	2347.80	6407.63	2306.00
	spatial	3363.00	7422.83	3280.00
FOREMAN	B-VOP	1542.60	2826.40	918.00
	scene	1151.20	2665.20	334.00
AKIYO	B-VOP	1540.20	2918.20	912.00
	scene	1023.60	2433.60	328.00

Looking at these run times, we observe an average performance gain between (TM-3) and (TM-2) of about 75%. Note that between (TM-3) and (TM-1) the performance gain is still about 46% on average. In particular, the XML transformation within the binary domain is up to 4 times faster than using plain text XML transformation tools and up to 9 times faster compared to (TM-2), i.e., BiM decoding, applying XSLT, and BiM encoding.

In the next section we will provide a detailed discussion of our results.

6. DISCUSSION

Our experiments have shown that the transformation of MPEG-21 (XML-based) metadata within the binary domain, as introduced in this paper, provides performance gains up to a factor of 4. These results can be explained due to the fact that our approach is based on binary bit pattern matching whereas traditional XML transformation tools use string comparisons which are quite cost-intensive.

The simple combination of XML transformation and BiM coding is very slow due to the en-/decoding overhead. However, following this approach the network load is reduced compared to transmission of plain text encoded XML documents. Furthermore, BiM enhances XML with streaming capabilities which can be exploited by corresponding transformation tools like STX.

The performance differences regarding the actual processing time between traditional XML transformation methods and our binary transformation approach becomes smaller, the more complex the structure of the metadata documents are. Thus, more AUs and FUUs are requires in order to transform the document within the binary domain which increases the complexity and explains this convergence. However, the experiments have emphasized that traditional XML transformations never reached the performance of the binary transformation.

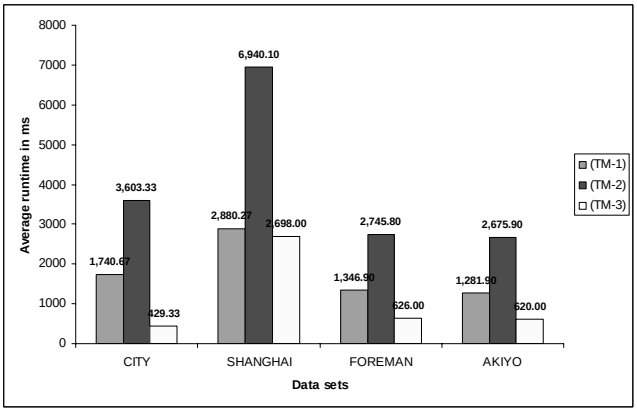

Figure 6 — Runtime comparison of the different transformations and test modes.

Another advantage of our transformation method can be found in the much smaller resource requirements regarding the memory of the processing machine. This results from the fact, that before applying any transformation, a legacy XSLT processor has to load the complete XML document into its memory which can cause serious resource problems when processing large files. Using our transformation method it is only necessary to read small parts of the (streamed) input document at the time.

Furthermore, the advantages of binary compression and the ability of streaming data can only be used in combination with such a binary representation of the metadata.

7. RELATED WORK

Traditional XML transformation tools have their focus on transforming only plain text encoded XML descriptions. XSLT [10] is its most popular representative which has the status of a W3C recommendation and many implementations and tools are available. One major drawback of XSLT is that the complete XML description must be in memory before being processed which is a burden in streaming scenarios. STX [11] which is based on SAX (Simple API for XML) events seems to be a promising candidate to overcome this burden but still operates on plain text XML. Tools like FXT (Functional XML Transformation) [12], XDuce [13], and HaXml [14] are not that established but operate on plain text as well.

Further related work can be found in [15] which provide an evaluation of several binary XML encoding optimizations, i.e., alternative serialization format, tokenization, and skip-to pointers. Note that BiM provides all these features. However, the evaluation in [15] concentrated on parsing performance only and has shown that such optimization facilitate a performance gain up to a factor of 6 (for parsing only) – depending on the document structure and the required information –

compared to the fasted XML parser they were aware of (i.e., xpat [16]).Finally, several articles such as [9] claim that BiM provides facilities for transforming XML data within the binary domain. In practice, however, at the time of writing this paper we were not aware of any paper dealing with this issue in detail. Therefore, we came to the conclusion that currently no work which is directly related to our work exists.

8. CONCLUSION AND FUTURE WORK

In this paper we presented a novel approach for transforming XML-based MPEG-21 metadata (gBSDs) within the binary domain. Furthermore, we evaluated our approach and compared it with traditional XML transformation techniques. While the results are very promising some (research) issues are still unsolved which lead to couple of future work items. The next steps include extending this approach to content-related timed XML-based metadata in general as well as improving the efficiency, e.g., by using optimized BiM software instead of the reference software, and decreasing the possibly large number of FUUs for very fine grained gBSDs by using an efficient payload parsers. Finally, we will work on the actual syntax and semantics of the application parameters which currently only available in an ad-hoc mode. Based on a subset of the XSLT specification and pre-defined style sheet templates for the identified most common operations we will work on a model and implementation which map these style sheet templates to appropriate BiM operations.

9. ACKNOWLEDGMENTS

This work was funded in part by the FWF (Fonds zur Förderung der wissenschaftlichen Forschung - Austrian Science Fund) under the project number P14789.

10. REFERENCES

[1] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting Multimedia Internet Content for Universal Access", *IEEE Trans. on Multimedia*, vol. 1, no. 1, Jan.-Mar. 1999, pp. 104-114.

[2] F. Pereira and I. Burnett, "Universal Multimedia Experiences for Tomorrow," *IEEE Signal Processing Magazine*, vol. 20, no. 2, Mar., 2003, pp. 63-73.

[3] M. Davis, S. King, N. Good and R. Sarvas, "From context to content: leveraging context to infer media metadata", *Proceedings of 12th annual ACM international conference on Multimedia*, New York, NY, USA, Oct. 2004, pp. 183-195.

[4] A. Vetro, C. Christopoulos, and T. Ebrahami, eds., *IEEE Signal Processing Magazine*, special issue on Universal Multimedia Access, vol. 20, no. 2, March 2003..

[5] I. Burnett et al., "MPEG-21: Goals and Achievements," *IEEE MultiMedia Magazine*, vol. 10, no. 6, Oct.-Dec. 2003, pp. 60-70.

[6] A. Vetro and C. Timmerer, "Digital Item Adaptation: Overview of Standardization and Research Activities", to appear in *IEEE Trans. on Multimedia*, vol. 7, no. 3, Jun. 2005.

[7] L. Böszörményi, H. Hellwagner, H. Kosch, M. Libsie, and S. Podlipnig, "Metadata Driven Adaptation in the ADMITS Project", *EURASIP Signal Processing: Image Communication Journal*, vol. 18, no. 8, Sep. 2003, pp. 749-766.

[8] K. El-Khatib, G. v. Bochmann, and A. El Saddik, "A QoS-Based Framework for Distributed Content Adaptation", *Proceedings of the IEEE International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE)*, Dallas, TX, Oct. 2004, pp. 300-303.

[9] U. Niedermeier, J. Heuer, A. Hutter, W. Stechele, and A. Kaup, "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, Aug. 2002, pp. 521–524.

[10] World Wide Web Consortium (W3C), "XSL Transformations (XSLT) Version 1.0", *W3C Recommendation*, Nov. 1999, available at http://www.w3.org/TR/xslt.

[11] "Streaming Transformations for XML (STX) Version 1.0", Working Draft, Jul. 2004, available at http://stx.sourceforge.net/.

[12] A. Berlea, H. Seidl, "Fxt - A Transformation Tool for XML Documents", *Proceedings of the Int'l XML Conference & Exposition*, Dec. 2001.

[13] H. Hosoya and B. C. Pierce, "XDuce: A typed XML processing language". *ACM Transactions on Internet Technology*, vol. 3, no. 2, May 2003, pp. 117-148.

[14] M. Wallace and C. Runciman, "Haskell and XML: Generic Combinators or Type-Based Translation?", *Proceedings of the Int'l Conf. on Functional Programming*, September 1999, pp. 148-259.

[15] R. J. Bayardo, D. Gruhl, V. Josifovski, J. Myllymaki, "An Evaluation of Binary XML Encoding Optimizations for Fast Stream Based XML Processing", *Proceedings of the 13th Int'l Word Wide Web Conf.*, New York, USA, May, 2003, pp. 345-354.

[16] C. Cooper, "Using expat", *xml.com*, 1999, available at http://www.xml.com/pub/a/1999/09/expat/index.html.