# PERFORMANCE OF A SCALABLE BITSTREAM ADAPTATION PROCESS BASED ON HIGH LEVEL XML DESCRIPTIONS

*Davy De Schrijver, Wim Van Lancker, and Rik Van de Walle*

Ghent University – IBBT, Department of Electronics and Information Systems – Multimedia Lab,
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium.
e-mail:{davy.deschrijver;wim.vanlancker;rik.vandewalle}@ugent.be

## ABSTRACT

Multi-channel publication environments are necessary in our future universal multimedia access world. Therefore, we must adapt our digital video content taking the restrictions and characteristics of networks and terminals into consideration. To make the adaptation possible in a flexible manner, we use a scalable bitstream which gives us the possibility to extract new bitstreams with other characteristics without the need of a complete decode-encode step. The adaptation must happen in a format-agnostic way and terminal-independent. To realize this goal, we describe our bitstream in XML by using the MPEG-21 BSDL standard. The adaptation is done in the XML domain instead of on the bitstream. The generation of the description and the new bitstream can take place automatically. In this paper, we describe the adaptation of a bitstream by using a description and the performance of this approach of adaptation.

## 1. INTRODUCTION

Nowadays, video plays an important role in our society (television, DVD, internet…) and will be more important in the future (thinking of possibilities of PDA, GSM…). Every video sequence must be encoded to make it possible to transport the video over a network, for example, digital television contains MPEG-2 encoded video sequences. The increasing amount of possible devices leads to a universal multimedia access where we want to create the content once and publish it on every possible device [1].

To publish the content on different devices and over different networks, containing dissimilar characteristics, the original (encoded) bitstream must be adapted. The adaptation can be done directly on the bitstream itself but in this paper, we will use a different approach. We describe the syntax of the bitstream in XML and do the adaptation in the XML domain. This gives us the opportunity to add extra information (metadata) to the (XML) description about the video sequence such as shot detection, semantic meaning of a scene… and to perform the adaptation based on the metadata.

The outline of the paper is as follows. First in section 2, we briefly discuss the generation of a bitstream description. Next in section 3, we explain a mathematical model, which predicts the generation time of an adapted bitstream. In section 4, we investigate the possible solutions for the XML transformations in the context of BSDL. The results of our experiments are summarized in section 5. Finally, section 6 concludes the paper.

## 2. HIGH-LEVEL DESCRIPTION OF A SCALABLE BITSTREAM

To enable the publication of content on heterogeneous devices utilizing different network characteristics, it is impractical to create content for every possible client configuration (such as screen resolution, bandwidth…). Therefore, we need a scalable bitstream, which gives us the possibility to keep only one parent bitstream and filter out different versions of the bitstream taken the enforced constraints into account.

Therefore, we have chosen to encode our video with the MC_EZBC algorithm [2]. This codec is wavelet-based and embedded scalable in quality, resolution and frame rate. To obtain these kinds of scalability, the codec uses a t+2D wavelet transform. This kind of codec consists of three phases (in contrast to the traditional two-step, encode-decode codecs). During the first phase, the codec encodes the video sequence and generates a near-lossless bitstream. From this bitstream, it is possible to extract other versions of the original bitstream during the second phase. These versions will have other characteristics like lower frame or bit rate. The last phase is typically the decode step of a bitstream.

Another important issue for this paper is that the adaptation could take place in every node of the delivery chain such as on the server; in a gateway or in an active router. Therefore, we need a universal adaptation engine and we will use the Digital Item Adaptation (DIA, [3]) part of the MPEG-21 framework [4]. To make the adaptation process format-agnostic (or format-independent), we describe the bitstream structure on a

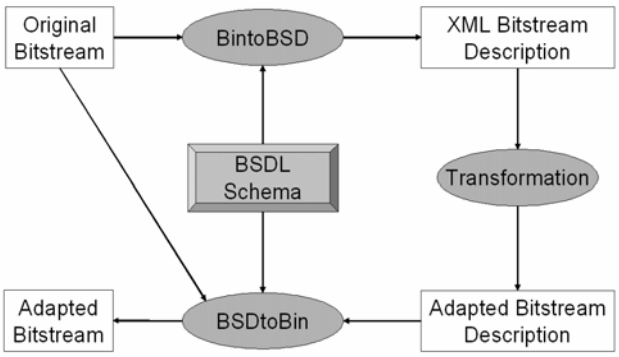


**Figure 1: BSDL Workflow**

higher level in XML (Extensible Markup Language) using BSDL (Bitstream Syntax Description Language, [5]) which is a part of DIA. BSDL is a language that is based on the W3C XML Schema language with some extensions to and restrictions on the W3C standard. In Figure 1, we see the workflow of the BSDL language. In this figure, we generate an XML description of a bitstream by using the BintoBSD tool. The original bitstream is embedded scalable and a BSDL schema represents the structure or syntax of the encoded bitstream. The generation can be done automatically. On these descriptions, a transformation can be executed to obtain the adapted descriptions. The transformation can be done by using an XSLT stylesheet, DOM or SAX parser. The transformation is the XML-equivalent adaptation step of the original bitstream. The last step in the BSDL workflow is the generation of the adapted bitstream from the adapted bitstream description. To make this generation possible in an automated way, we need the description together with the BSDL schema and the original bitstream. The BSDL schema is necessary to understand the syntactical meaning of the different tags in the description and the original bitstream is required during this process because the description contains references to this bitstream.

To make this approach practically usable, it is necessary to perform the adaptation of the description together with the generation of the (adapted) bitstream in real time. In the following sections, we will discuss the performance of the different steps.

## 3. MATHEMATICAL MODEL OF THE BITSTREAM GENERATION

The last step in the BSDL approach is the generation of the new bitstream from a description. The first place to adapt the bitstream is in most cases the (content) server because the (original) parent near-lossless bitstream will be too big to send over a network. So, it is necessary to predict the execution time of the BSDtoBin tool to avoid

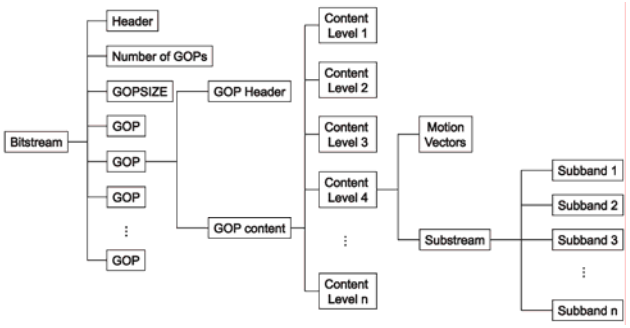


**Figure 2: Structure of a bitstream description**

server overloading. A mathematical model can be used as predictor of the execution time.

In Figure 2, we see the general structure of a bitstream description that is used as input document for the BSDtoBin tool. We must remark that the *header* and *subband* elements contain metadata about the bit planes of the encoded sequence. This is necessary to make the SNR scalability possible and is not encapsulated in the bitstream. The bitstream generator (BSDtoBin tool) must parse that kind of descriptions and the execution time of the generation depends on the sequence characteristics (such as number of frames, length of GOP structures, frame resolution…) and terminal properties (such as CPU speed, hard disk, available memory…). Our model must be sequence and terminal independent so that it is usable on every possible device and for every (MC_EZBC encoded) video sequence. In our model, the terminal characteristics are called the parameters of the model and the sequence properties are called the variables.

Suppose, as parameters:

a = number of temporal levels
b = number of spatial levels
c = number of GOP structures
d = number of bitplanes

and as variables:

u = constant startup time
v = time to parse a subband
w = time to parse a motion vector
x = time to parse a GOP header
y = time to parse the metadata

then the model is:

$$Total\ execution\ time = u + parse\ BSD + IO$$

$$= u + \sum_{s=1}^{c} (parse\ GOP_s) + IO$$

$$= u + \sum_{s=1}^{c} \left( GOPHeader_s + \sum_{r=0}^{a-1} content\ level_{r_s} \right) + IO$$

$$= u + \sum_{s=1}^{c} \left( x + cont\ level_{0_s} + \sum_{r \neq 0}^{a-1} \left( MV_r + subbands_r \right) \right) + IO$$

| Sequence | # frames | resolution | Size bitstream (KB) | XSLT (s) | Java – DOM (s) | Java – SAX (s) | C++ - DOM (s) | C++ - SAX (s) |
|---|---|---|---|---|---|---|---|---|
| Seq 1 | 40 | 352 x 240 | 2425 | 155.7 | 1.8 | 0.6 | 0.5 | 0.4 |
| Seq 2 | 121 | 352 x 288 | 6474 | 844.0 | 4.8 | 1.6 | 1.4 | 1.0 |
| Seq 3 | 300 | 352 x 288 | 14358 | 2023.0 | 8.7 | 3.5 | 2.8 | 2.3 |
| Seq 4 | 541 | 352 x 240 | 33349 | > 2100.0 | 12.7 | 4.8 | 4.4 | 3.2 |
| Seq 5 | 528 | 1280 x 720 | 115174 | > 2100.0 | 13.7 | 6.7 | 5.1 | 3.9 |
| Seq 6 | 529 | 1280 x 720 | 188282 | > 2100.0 | 13.8 | 6.7 | 5.1 | 3.9 |

**Table 1: Performance of the temporal transformation by using different technologies**

$$= u + \sum_{s=1}^{c} \left( x + b\left(v + dy\right) + \sum_{r \neq 0}^{a-1} \left( w2^{r-1} + b\left(v + dy\right) \right) \right) + IO$$

$$= u + c\left( x + b\left(v + dy\right) + \sum_{r \neq 0}^{a-1} \left( w2^{r-1} + b\left(v + dy\right) \right) \right) + IO$$

To predict the execution time of the BSDtoBin tool, we must evaluate the above model. The parameters can be filled in once we know the sequence and the variables are constant values for every terminal. To obtain these values, we determine the constants during the installation phase of the software by using specific formulated descriptions, for example a description with only motion vector information.

## 4. ADAPTATION PROCESS

The goal of BSDL is to generate adapted bitstreams without manipulating the bits of the bitstream themselves. Every bitstream is embedded temporal, spatial and quality scalable. These kinds of scalability must be exploited in the XML domain by simple editing operations.

The first type of scalability is a temporal reduction by a power of two. To obtain that kind of adapted descriptions, we must eliminate the highest content levels of each GOP in Figure 2. The second transformation executes the spatial transformation. Therefore, we must delete the highest subbands of each content level. Finally, the SNR scalability cannot be performed by using the information contained in the bitstream, as we can see in Figure 2. During the SNR transformation, we must truncate every subband in such a manner that we respect the given global bit rate and that the quality is constant and as high as possible. We need information about the composition of the different subbands, in particular the number and the length of the different bit planes that a subband contains. The extra information about the bit planes is the encapsulated metadata as used in section 3. Based on general metadata (contained in the header tag), we can calculate the bit plane where we must truncate the subbands. Once we know this bit plane, we need the subband-specific metadata (contained in the subband tags of Figure 2) to know the length of the bit plane in the current subband and where to prune the subband.

How to implement these transformations is not standardized in the MPEG-21 DIA standard. The decision lies in the hands of the adaptation engine implementer. We have compared different possible technologies qua performance in execution time and memory consumption. We have implemented the different transformations in XSLT, Java and C++ by using a DOM and SAX library.

To perform the measurements, we have run every transformation in every technology 5 times and calculated the average over the 5 runs. The measurements were done on a PC having an Intel Pentium IV CPU, clocked at 2.8GHz with Hyper-Threading and having 1GB of RAM at its disposal. The used operating system was Windows XP Pro (service pack 2) and Sun Microsystems's Java 2 Runtime Environment (Standard Edition version 1.4.2_04) was running as JVM.

## 5. EXPERIMENTAL RESULTS

First, we discuss the performance of the transformation implementations. We could give an overview for the three scalabilities, but in this paper, we only discuss the temporal transformation by a reduction of two temporal levels. The same conclusions of analysis can be conducted for the other transformations. In Table 1, we give an overview of the measurements for different sequences and implementations. We have selected six sequences with different lengths and resolutions as reflected in the table. During the encode phase, we have used GOP structures of 16 frames. In this table, we see that XSLT is completely unusable as transformation engine in the context of bitstream description adaptation. The difference between a Java and C++ implementation by using a DOM library is significant; the origin of the difference lies in the fact that Java needs more time to construct the internal DOM tree in memory and to serialize the tree to a file. The transformation of the (internal) tree is equally fast in both situations. Finally, we see that the difference between the SAX implementation in Java and C++ is smaller than in case of the DOM implementation.

| Sequence | Size Bitstream (KB) | 1st run (s) | 2nd run | | |
|---|---|---|---|---|---|
| | | | Parsing time (s) | I/O time (s) | Total time (s) |
| Seq 1 | 2425 | 0.70 | 0.25 | 0.03 | 0.28 |
| Seq 2 | 6474 | 1.36 | 0.87 | 0.02 | 0.89 |
| Seq 3 | 14358 | 2.53 | 1.97 | 0.03 | 2.00 |
| Seq 4 | 33349 | 3.67 | 2.74 | 0.26 | 3.00 |
| Seq 5 | 115174 | 6.93 | 3.00 | 3.45 | 6.45 |
| Seq 6 | 188282 | 10.28 | 3.09 | 6.71 | 9.80 |
| Seq 6 after temporal reduction | 67675 | 5.03 | 1.89 | 2.58 | 4.47 |

**Table 2: Performance of the BSDtoBin tool**

Another measurement is the memory consumption during the transformation. For DOM-based implementations, we need 180MB for the last three sequences and only 2MB for the SAX-based approach. The longer the sequence, the more memory we need for the construction of the internal DOM tree. In contrast with SAX, where we always need 2 MB independent of the length of the sequence. Therefore, we can conclude that a SAX-based implementation is the only usable technique for a description transformation when the sequences are considerably long.

Secondly, we have measured the performance of the BSDtoBin tool. During the execution of the application, we noticed that the time that was spent on I/O was significant. Therefore, we have calculated the time that was spent on parsing and on I/O by using our model of section 3. For each tested description, we have executed two runs immediately after each other without closing the application. The results of the second run are the most important because the JIT-compiler of JVM has done his work and all the classes are loaded in the memory. In a use case, the BSDtoBin application on a server will never close and will be waiting on new requests while the (compiled) classes are still in the memory.

In table 2, we see the results of the BSDtoBin tool during the 1st and 2nd run. The difference between the two execution times is in all situations approximately 0.5s (the time to load and compile most classes and to construct an internal DOM tree of the BSDL schema). Further in the table, we see that the time spent on I/O is high for the high definition sequences. Finally, we have measured the execution time of a transformed description because these descriptions will be used in a real use case.

When we combine the results from table 1 and table 2, we see that the transformation of the description and generation of the adapted bitstream is possible in real time. So, we can use BSDL in streaming use cases.

## 6. CONCLUSIONS

In this paper, we have discussed how we can adapt scalable bitstreams by using XML descriptions. To realize this aim, we have used the MPEG-21 BSDL standard. The adaptation step in the XML domain is a transformation of an XML document. We have seen that the usage of a SAX parser is the only acceptable solution for the transformation of a bitstream description. To handle multiple requests, we must know the execution time of the bitstream generator (BSDtoBin). Therefore, we have formulated a mathematical model and by using this model, we have seen that the generation of a new bitstream can be done in real time.

## 7. ACKNOWLEDGMENTS

The research activities that have been 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 Office for Scientific, Technical and Cultural Affairs (OSTC), and the European Union.

## 8. REFERENCES

[1] A.Vetro, C. Christopoulos, T. Ebrahimi "Universal Multimedia Access", *IEEE Signal Processing magazine*, 20 (2) 16-16, March 2003

[2] P.Chen, J.W. Woods, "Fully Scalable Subband / Wavelet Coding",Rensselaer Polytechnic Institute, Troy New York, May 2003

[3] Moving Picture Experts Group, "Text of ISO/IEC 21000-7 FCD – Part 7: Digital Item Adaptation," ISO/IEC JTC1/SC29/WG11 N5845, July 2003

[4] I. Burnett, R. Van de Walle, K. Hill, J. Bormans, and F. Pereira "MPEG-21: Goals and Achievements", *IEEE Multimedia,* IEEE Computer Society, pp. 60-70, 2003

[5] Myriam Amielh, Sylvain Devillers, "Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content Adaptation," WWW2002, May 2002