AN EVALUATION OF EXISTING METADATA COMPRESSION AND ENCODING TECHNOLOGIES FOR MPEG-21 APPLICATIONS

Christian Timmerer, Ingo Kofler, Johannes Liegl, and Hermann Hellwagner

Department of Information Technology (ITEC), Klagenfurt University, Klagenfurt, Austria
{christian.timmerer, hermann.hellwagner}@itec.uni-klu.ac.at, {ikofler, jliegl}@edu.uni-klu.ac.at

Department of Information Technology (ITEC)
Klagenfurt University
Technical Report No. TR/ITEC/05/1.12
October 2005

AN EVALUATION OF EXISTING METADATA COMPRESSION AND ENCODING TECHNOLOGIES FOR MPEG-21 APPLICATIONS

Christian Timmerer[‡]*, Ingo Kofler, Johannes Liegl, and Hermann Hellwagner*

Department of Information Technology (ITEC), Klagenfurt University, Klagenfurt, Austria
{christian.timmerer, hermann.hellwagner}@itec.uni-klu.ac.at, {ikofler, jliegl}@edu.uni-klu.ac.at

ABSTRACT

XML-based metadata for digital media is becoming increasingly important, as a consequence also calling for efficient encoding and compression schemes for the storage and transport of this metadata. Moreover, support for streaming the XML metadata in conjunction with the media data is highly desirable. Such support is provided, for instance, by MPEG's Binary Format for Metadata (BiM) encoding approach, which facilitates fragmenting, delivering, and accessing the metadata in so-called Access Units (AUs). In this paper, we present a quantitative evaluation of existing XML metadata compression and encoding techniques, reaching from widely used state-of-the-art data compression algorithms to sophisticated XML-aware encoding schemes. The comparison is based on compressing MPEG-21 generic Bitstream Syntax Descriptions (gBSDs) which can grow to non-negligible sizes. The main conclusion from this investigation is that in terms of pure compression efficiency on XML files, the BiM approach (exemplified by the MPEG reference software as well as a commercial version thereof) is comparable – in terms of performance – with traditional data or specific XML compression tools. However, when XML metadata have to be fragmented, compressed, and streamed in such fragments, the results indicate that the BiM approach is superior to the other schemes.

1. INTRODUCTION

XML-based metadata is becoming increasingly important and has been adopted by various communities including the digital media computing community [1]. Within the Moving Picture Experts Group (MPEG), two work items deal with metadata, namely the "Multimedia Content Description Interface" (MPEG-7) [2] and the "Multimedia Framework" (MPEG-21) [2]. In both cases, W3C's XML Schema has been adopted or extended as needed. However, in many use cases, e.g., within W3C, the transport of metadata was neither foreseeable nor desirable. This changed with the emergence of issues in a context that is generally referred to as Universal Multimedia Access (UMA) [4][5]. In UMA, metadata is getting a key role and is used to support seamless access to any type of (multimedia) content anywhere and anytime. These issues have been recognized by MPEG which resulted in the introduction of the MPEG-21 standard. MPEG-21 aims at enabling transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities [3].

A vital part and important with regard to UMA is MPEG-21 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. Device independence is realized through a unified description format providing means for describing the usage environment of Digital Items, such as terminal and network characteristics. Coding format independence is achieved by utilizing metadata describing the syntax of a media bit-stream in a generic way; in addition, the constraints and restrictions imposed by both the usage environment and the content provider can be expressed in a normative way and taken into account for content adaptation and delivery.

As all related MPEG-7/-21 standards, DIA also makes use of XML. Additionally, since DIA can be applied within all stages of the multimedia delivery chain, i.e., from the provider to the consumer, it becomes apparent that metadata needs to be transmitted (together with the actual media data) over the network, e.g., for allowing metadata-driven content adaptation within intermediate network devices such as proxies or gateways. Due to the metadata's plain text nature, it is obvious that more efficient transport encoding schemes for XML-based metadata are desirable, especially when streaming requirements to constrained devices are not negligible or compared to highly efficient coding formats of its corresponding streaming media data, e.g., MPEG-4 AVC/H.264 [7]. Therefore and in order to overcome the

[‡] christian.timmerer@itec.uni-klu.ac.at; phone +43 (463) 2700 3621; fax +43 (463) 2700 3699; www.itec.uni-klu.ac.at

verbosity of plain text encoded XML as well as its non-existing streaming capabilities, we argue that appropriate compression, encoding, or even an alternative serialization format for XML-based metadata is required.

In this paper we present a quantitative evaluation of existing XML metadata compression and encoding techniques, reaching from widely used state-of-the-art data compression algorithms to sophisticated XML-aware encoding schemes. The aim of this evaluation is to investigate how well specific MPEG-7/-21 metadata serialization schemes (BiM) work as compared to general text and XML compression tools. Furthermore, the comparison should provide guidelines for the readers for selecting the suitable technology for their applications. We emphasize that support for streaming the XML metadata is increasingly important in distributed multimedia systems.

The remainder of this paper is organized as follows. In Section 2 we describe the compression and encoding approaches that we consider and compare. The experimental setup and results are given in Sections 3 and 4, respectively. A discussion of the results is given in Section 0 and Section 6 concludes the paper.

2. COMPRESSION AND ENCODING TECHNIQUES FOR XML-BASED METADATA

In this section we provide an overview of some representative compression and encoding techniques for XML-based metadata. First we will briefly describe traditional data compression approaches before providing details about so-called intrinsic XML encoding approaches.

2.1. Traditional Data Compression Techniques

bzip2. This compression technique is based on the Burrows-Wheeler block-sorting lossless data compression algorithm as described in [8]. The input is processed in blocks of a certain size, which can be adjusted to either make the compression faster or to increase the compression ratio. Each of the blocks is transformed with the so called Burrows-Wheeler Transformation (BWT) which coarsely consists of building all possible rotations of the block's content by cyclic shifts and sorting them. The advantage of this reversible transformation is that the result can be efficiently processed with fast locally-adaptive compression algorithms, like a move-to-front coder [9] in combination with a Huffman coder [10], which are used in bzip2. The compression achieved with bzip2 is comparable with those of good statistical modelers (e.g., [8]), but with the advantage that the compressing speed is close to the compression based on the Lempel/Ziv (LZ) algorithms [11][12].

gzip, WinZip. Both compression tools use the deflate algorithm which incorporates a variation of the LZ77 compression algorithm and Huffman coding. The LZ77 algorithm reuses already seen strings of a message for the transmission of upcoming ones. The longest possible substring is searched in the already processed part of the message and only its starting position and length are transmitted. The search for a matching substring is restricted to the last processed part of the message. This makes the algorithm automatically adaptive to a change of the probability distribution in the message. Given a probability distribution of the source symbols the Huffman algorithm calculates a code table which assigns every symbol a sequence of bits. The Huffman code is an optimal prefix code which means that no other assignment of bit sequences would achieve a better compression and that no delimiter between two Huffman coded symbols is needed. In the deflate algorithm, the LZ77 is used to identify matches and replace them with (start position, length) pairs. The matches themselves are compressed with one Huffman tree and match distances are compressed with another tree.

2.2. XML-Aware Encoding Schemes

XMLPPM. XMLPPM is a compression tool for XML documents that combines the well known Prediction by Partial Match (PPM) and the Multiplexed Hierarchical Modeling (MHM) algorithms [13][14].

Generally, PPM compression models maintain statistics concerning which symbols have been seen in which contexts of preceding symbols. For each input symbol, the model is used to determine a probability range. This range is then used to transmit the symbol using arithmetic coding. During the compression the coder updates its statistics continuously which is also done at the decoder side.

In order to improve the efficiency of the PPM compression, MHM is used, which is aware of the structure of an XML document. The algorithm uses four different PPM compression models: one for element and attribute names, one for element structure, one for attribute values, and one for strings (i.e., the actual data within the XML elements). All are multiplexed based on the syntactic context delivered by the parser. The idea of switching between the four models improves the overall compression but has still its drawback, because multiplexing breaks existing sequential dependencies between elements, attributes, etc. For example, the information that an enclosing XML element has a strong dependency with the enclosed data would not be used for improving the PPM compression. Thus, MHM uses a second technique which improves the efficiency by providing the information to the different underlying PPM compressors in which context (i.e., XML element) the actual symbol is encountered.

XMill. XMill [15] exploits the self describing nature of XML for compression. In order to achieve this goal it

leverages existing compression algorithms and tools like zlib (the library function version of gzip) and some simple data type specific compressors.

XMill applies three principles to compress XML data. First, structure and data are separated. This means that XML elements and attributes (the structure) are compressed separately from the actual data items (i.e., strings) and attribute values. Second, related data items are grouped into containers which are compressed separately. Third, the last principle is to apply semantic processors on the containers depending on the actual type of content, e.g., optimized for strings or numbers. The information which semantic processor is applied on a container is obtained from the container expression provided by the user through the command line interface. If no container expression is provided the text compressor which just copies the data into the containers is applied to every container. Together with the path of the data value (i.e., the sequence of XML elements from the root to the data value) in the XML document the container expression is used to determine the container in which the data item is filled. All containers are kept in a fixed-size memory buffer. If this buffer is full, zlib is applied and it is stored to disk and the compression resumes.

XMill is not designed to work with a query processor. The target applications are data exchange to better utilize network bandwidth (whole documents only) and data archiving to reduce space requirement.

2.3. A Binary Format for Metadata

MPEG's Binary Format for Metadata (BiM) [17] 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. To that end, 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 as follows:

- The FU 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 FU context is used to uniquely determine the location of the fragment in the XML document.
- The FU payload contains the actual XML data according to the context.

By definition, each AU can be decoded separately while ensuring validity against the corresponding XML Schema. The FUUs are processed according to the FU command, i.e., added to, deleted from, or replaced in 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.

Table 1 — XML compression and encoding tools.

Tool Name	Version	Program. Language	Platform
bzip2	1.0.2	C/C++	Windows, Unix, Linux
WinZip, gzip	9.0 / 1.3.5	C/C++	Windows, Unix, Linux
XMLPPM	0.98.2	C/C++	Windows (compiled under MS Visual Studio v. 6.0 with expat v. 1.95.8 and libiconv 1.8)
XMill	0.8	C/C++	Windows
MPEG-7 BiM reference software	Feb. 2005	Java	Windows
BinXML™	3.0.1	Java/C++	Windows, Linux

3. EXPERIMENTAL SETUP

We conducted a series of experiments to measure and compare the performance of the different metadata compression and encoding tools. In particular, we measured the decoding speed as well as the achieved compression ratio of the different tools. In this section, we provide an overview of the test data set and how we conducted the experiments. All tests were performed on a 1.6 GHz P4 Mobile machine with 1024 MB main memory and Windows XP installed. An overview of the considered XML compression and encoding tools is given in Table 1. Note that BinXML is a commercial version of, and largely compliant to, the BiM.

The test data set comprises generic Bitstream Syntax Descriptions (gBSDs) describing audio, video, and image resources. For audio the MPEG-4 Bit Sliced Arithmetic Coding (BSAC) codec is chosen providing fine-grained scalability through enhancement layers. The BSAC gBSD is described at a frame and group of frames (GOF) level (i.e., 10 and 25 frames per GOF respectively). For video we used MPEG-4 Visual Elementary Streams (VES) encoded at the Advanced Simple Profile which includes B-frames. The VES gBSDs are provided at the same granularity as the BSAC gBSDs. Note that MPEG-4 introduced object-based coding and therefore the equivalent to frames is called Video Object Planes (VOPs). Finally, images are encoded using the JPEG 2000 wavelet-based compression algorithms. The JPEG 2000

```
<dia:DIA xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
    xmlns:bs1="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS">
  <dia:DescriptionMetadata>
   <dia:ClassificationSchemeAlias alias="M4V" href="urn:mpeg:mpeg4:visual:cs:syntacticalLabels"/>
  </dia:DescriptionMetadata>
  <dia:Description xsi:type="gBSDType" addressUnit="byte" addressMode="Absolute"
       bs1:bitstreamURI="lotr.cmp">
    <gBSDUnit syntacticalLabel=":M4V:VO" start="0" length="18"/>
    <gBSDUnit syntacticalLabel=":M4V:GOV" start="18" length="105947" marker="0">
      <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 ...-->
    </gBSDUnit>
    <!--... and so on ...-->
  </dia:Description>
</dia:DIA>
```

Document 1 — gBSD fragment describing an MPEG-4 Visual Elementary Stream encoded with Advanced Simple Profile and 10 Visual Object Planes (VOPs) per Group of VOPs (GOV).

images are completely described by the gBSD, i.e., without providing the gBSD data on a fragment basis.

Document 1 shows a fragment of the gBSD describing the MPEG-4 VES including the DIA wrapper elements and Document 2 provides the gBSD information for a single MPEG-4 BSAC encoded audio frame. The DIA wrapper element is similar to that of Document 1.

4. RESULTS

The results of the experiments are depicted in Figure 1 through Figure 5. For XMill we have provided two measures, i.e., one using the PPM codec and another one using the bzip2 codec for data compression. Figure 1 shows the results for the gBSD describing an MPEG-4 VES in terms of the compression ratios of the tools with respect to the gBSD plain text counterpart. Figure 2 shows the metadata overhead in percent with respect to the original media resource size. Similar results can be found in Figure 3 and Figure 4 for the gBSD describing an MPEG-4 BSAC stream. All these results are measured in "Total" (i.e., the complete gBSD is compressed/encoded) or on a per fragment basis (i.e., individual frames, VOPs, GOFs, or GOVs are compressed/encoded). In the latter two cases we differentiated between 10 (i.e., GOV10 or GOF10) and 25 (i.e., GOV25 or GOF25) frames per GOF or VOPs per GOV, respectively. Note that only BiM and BinXML provide such kind of fragmentation and streaming functionality for XML-based metadata. For all other compression and encoding tools, these fragments are represented by using the textual equivalent of BiM and are subsequently compressed with the respective tool.

On the one hand, the results of the achieved compression ratio clearly show that the so-called XML-aware compression tools provide the best results when compressing the complete gBSD compared to traditional

```
<gBSDUnit start="0" length="2680">
 <gBSDUnit length="208" addressMode="Consecutive">
  <Parameter length="11">
   <Value xsi:type="b11">335</Value>
  </Parameter>
  <gBSDUnit length="5"/>
  <Parameter length="6">
   <Value xsi:type="b6">48</Value>
  </Parameter>
  <gBSDUnit length="186"/>
 </gBSDUnit>
 <gBSDUnit length="2472" marker="el0-0el1-8el2-
24el3-32el4-48el5-56el6-64el7-80el8-88el9-104el10-
112el11-128el12-136el13-152el14-160el15-168el16-
184el17-192el18-208el19-216el20-232el21-240el22-
256el23-264el24-280el25-288el26-296el27-312el28-
320el29-336el30-344el31-360el32-368el33-384el34-
392el35-408el36-416el37-424el38-440el39-448el40-
464el41-472el42-488el43-496el44-512el45-520el46-
528el47-544el48-2472e" addressMode="Consecutive"/>
</gBSDUnit>
```

Document 2 — gBSD excerpt describing an MPEG-4 BSAC frame. The last gBSDUnit element contains a marker providing a handle for adapting the actual content.

compression schemes. However, in case where only fragments of the media resource are described, BiM and BinXML reach similar or even much better compression ratios than traditional or XML-aware compression and encoding schemes.

On the other hand, the metadata overhead significantly increases when fragmenting the XML in a very fine-grained manner, i.e., at a frame or VOP level, and encoding or compressing it. Clearly, the metadata overhead increases with decreasing fragment size.

For JPEG2000 (not shown here due to space limitations), the plain text metadata are about 70% of the size of the media data. All the tools are comparably successful in reducing this overhead to 2–3% by means of compression (compression ratio of approx. 25 to 35), except BiM reference software (compression ratio 18).

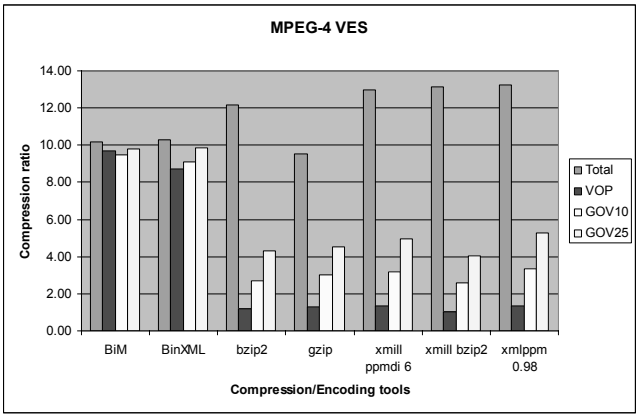

Figure 1 — Compression ratio for gBSD describing MPEG-4 VES.

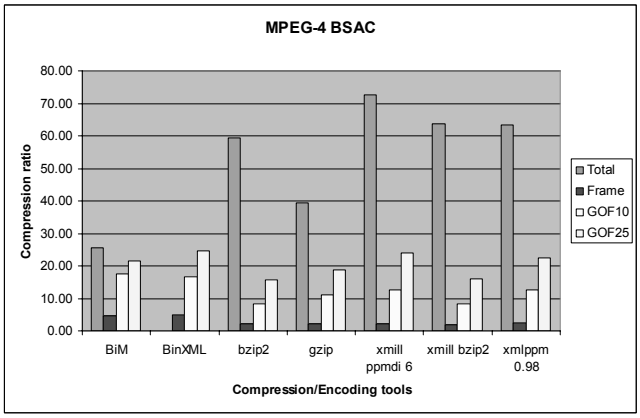

Figure 3 — Compression ratio for gBSD describing MPEG-4 BSAC.

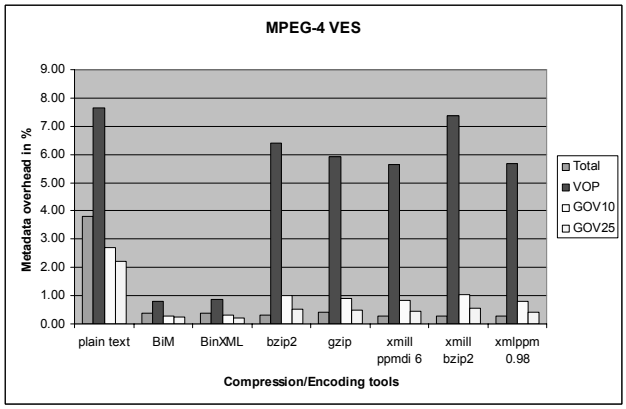

Figure 2 — Metadata overhead for gBSD describing an MPEG-4 VES with regard to the size of the media resource.

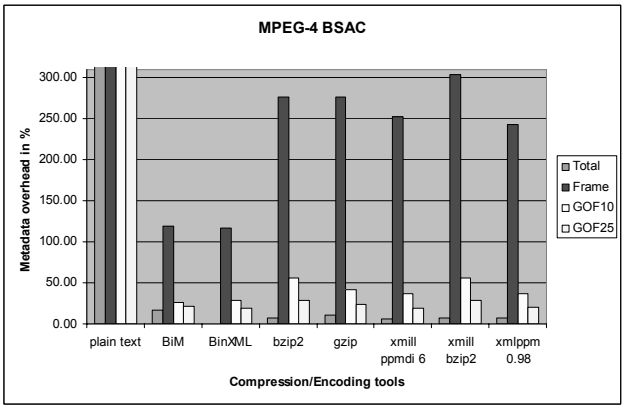

Figure 4 — Metadata overhead for gBSD describing an MPEG-4 BSAC with regard to the size of the media resource.

Finally, Figure 5 provides runtime measurements for decoding the metadata which shows that the BiM reference software is much slower than the other tools. The chart has been scaled down for better readability, i.e., BiM decoding (with the reference software) lasts around 4,000 ms for the "MPEG-4 VES" test data stream. In the next section we will discuss these results in detail.

5. DISCUSSION

The runtime measurements clearly show that BiM reference software is slower than the other tools. However, we kindly note that, first, for BiM only the MPEG reference software was provided which targets only functionality and not performance. Second, for the reference software only a Java version is available whereas other tools are written in C/C++. However, the commercial version thereof, i.e., BinXML, shows a significant improvement over the reference software as exemplarily shown for the MPEG-4 VES in Figure 5. Please note that the purpose of the BinXML decoder is to generate events (i.e., similar to SAX) and therefore speed up the overall application process, e.g., interpreting or transforming the XML data, and not to generate XML

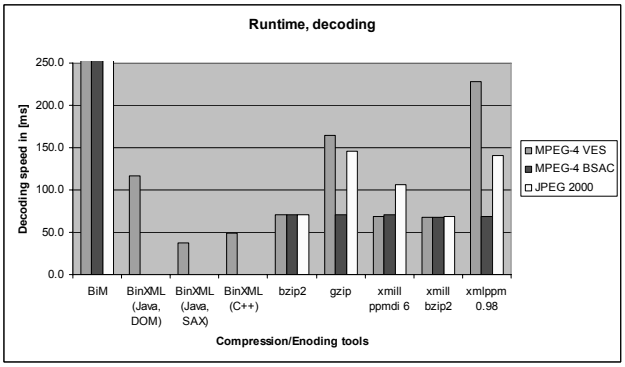

Figure 5 — Runtime measurements.

documents. Therefore, the measurements only provide the decoding speed – including the generation of the SAX events – without the actual XML document generation which would quadruple the decoding runtime. All other runtime measurements include the XML document generation – without generation of SAX events.

From the compression ratio point of view, we conclude that, for streaming XML-based metadata, BiM provides comparable results to traditional compression

tools and in some cases, i.e., MPEG-4 VES, the compression ratios are actually much higher than others. Furthermore, we observe that the larger the fragments are (i.e., the XML data to be encoded into one access unit) the higher the compression ratio and the smaller the metadata overhead. However, in case the XML fragments reach the size of the whole XML document (gBSD), traditional compression and encoding tools provide better results. Therefore, the BiM approach is not suitable for simple compression of whole documents without exploiting its streaming functionality. Only the combination of streaming XML and appropriate compression schemes provides satisfactory results which can be achieved by following the BiM approach. If the application requirements focus on storage only, i.e., without streaming the metadata over the network, we propose the usage of traditional compression or to a certain extent XML-aware compression techniques.

Regarding the metadata overhead the results have shown that encoding the metadata in a fine-grained manner is not advisable. Additionally, the achieved compression ratio confirms this statement. However, if such a kind of fragmentation is needed, the BiM approach should be chosen which results in the smallest overhead and also reasonable compression ratios.

6. CONCLUSION

We presented a quantitative evaluation of existing XML metadata compression and encoding techniques, using MPEG-21 gBSDs as test data. The main conclusion from this investigation is that in terms of pure compression efficiency on XML files, the BiM approach (exemplified by the MPEG reference software as well as a commercial version thereof) is comparable – in terms of performance – with traditional data or specific XML compression techniques (tools). However, when XML metadata have to be fragmented, compressed, and streamed in such fragments (i.e., encoded and not only compressed), the results indicate that the BiM approach is superior to the other schemes. This encourages further research into streaming and processing MPEG-7/-21 metadata based on this approach, which is required, e.g., for distributed adaptation of multimedia contents.

7. ACKNOWLEDGMENTS

The authors would like to thank Expway for providing BinXML™ and their technical support during the course of the experiments.

8. REFERENCES

[1] F. Nack, "The Future in Digital Media Computing is Meta," *IEEE MultiMedia Magazine*, vol. 11, no. 2, Apr.-Jun. 2004, pp. 10-13.

[2] S.-F. Chang, A. Puri, T. Sikora, and H. Zhang, eds., *IEEE Trans. on Circuits and Systems for Video Technology, special issue on MPEG-7*, vol. 11, no. 6, 2001.

[3] F. Pereira, J. R. Smith, and A. Vetro, eds., *IEEE Trans. on Multimedia, special section on MPEG-21*, vol. 7, no. 3, Jun. 2005.

[4] 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.

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

[6] A. Vetro and C. Timmerer, "Digital Item Adaptation: Overview of Standardization and Research Activities", *IEEE Trans. on Multimedia*, vol. 7, no. 3, Jun. 2005, pp. 418-426.

[7] G. Sullivan and T. Wiegand, "Video Compression - From Concepts to the H.264/AVC Standard," *Proc. of the IEEE, Special Issue on Advances in Video Coding and Delivery*, Vol. 93, No. 1, Jan. 2005, pp. 18-31.

[8] M. Burrows and D.J. Wheeler, "A block-sorting lossless data compression algorithm," *Technical report*, DEC, May 1994; http://citeseer.ist.psu.edu/76182.html.

[9] J.L. Bentley, D.D. Sleator, R.E. Tarjan, and V.K. Wei, "A locally adaptive data compression algorithm", *Communications of the ACM*, vol. 29, no. 4, Apr. 1986, pp. 320–330.

[10] D. A. Huffman, "A Method for the Construction of Minimum Redandancy Codes", *Proc. of the Institute of Radio Engineers 40*, Sep. 1952, pages 1098–1101.

[11] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression", *IEEE Transactions on Information Theory*, vol. IT-23, no. 3, May 1977, pp. 337–343.

[12] J. Ziv and A. Lempel, "Compression of individual sequences via variable rate coding", *IEEE Trans. on Information Theory*, vol. IT-24, no. 5, Sep. 1978, pp. 530–535.

[13] J. Cheney, "Compressing XML with Multiplexed Hierarchical PPM Models," *Proc. of IEEE Data Compression Conf.*, Snowbird, Utah, USA, Mar. 2001, pp. 163–172.

[14] XMLPPM homepage; http://xmlppm.sourceforge.net/.

[15] H. Liefke and D. Suciu, "XMill: an Efficient Compressor for XML Data," *Proceedings of the 2000 ACM SIGMOD Int'l Conf. on Management of Data*, Dallas, USA, May, 2000, pp. 153-164.

[16] Jun-Ki Min, Myung-Jae Park, Chin-Wan Chung, "XPRESS: A Queriable Compression for XML Data," *Proc. 2003 ACM SIGMOD Int'l Conf. on Management of Data*, San Diego, USA, Jun., 2003, pp. 122-133.

[17] U. Niedermeier, J. Heuer, A. Hutter, W. Stechele, and A. Kaup, "An MPEG-7 tool for compression and streaming of XML data", *Proc. of the 2002 IEEE Int'l Conf. on Multimedia and Expo (ICME)*, vol. 1, Lausanne, Switzerland, Aug. 2002, pp. 521–524.