Transport Mechanisms for Metadata-driven Distributed Multimedia Adaptation[1]

Michael Ransburg, Christian Timmerer, and Hermann Hellwagner
Department of Information Technology (ITEC)
Klagenfurt University, Klagenfurt, Austria
{michael.ransburg, christian.timmerer, hermann.hellwagner}@itec.uni-klu.ac.at

Abstract—**The information revolution of the last decade has resulted in a phenomenal increase in the quantity of multimedia content available to an increasing number of different users with different preferences who access it through a plethora of devices and over heterogeneous networks. In order to address the amount of different content types, MPEG-21 Digital Item Adaptation (DIA) introduces interoperable description tools which enable coding format independent adaptation. Bandwidth-efficient transport of the content to terminals with different capabilities and through a variety of access networks with various characteristics requires adaptation facilities not only on the server but also within the network. In this paper we present transport mechanisms for MPEG-21-based metadata enabling generic adaptation within the network. Three different transport mechanisms for delivering this metadata in conjunction with the corresponding multimedia content are evaluated and a payload format for the transport of this metadata is presented. Furthermore, we performed measurements which demonstrate the bandwidth benefits of our distributed adaptation approach compared to server-centric adaptation in a multicast scenario. Finally, we applied various encoding formats for the metadata which further reduces the metadata overhead.**

Keywords—MPEG-21 Digital Item Adaptation; distributed multimedia adaptation; metadata transport

I. INTRODUCTION

In today's multimedia content delivery architecture, adaptation becomes more and more important. Content providers aspire towards serving a plethora of heterogeneous end devices and networks without neglecting economical principles, i.e., if multiple versions of the same content are maintained. Therefore, a single high quality multimedia resource is stored on the server and adapted according to the usage environment on demand. The MPEG-21 Digital Item Adaptation (DIA) standard [1][2] specifies normative description tools enabling the construction of device and coding-format independent adaptation engines in an interoperable way [3]. However, it has been argued that it is not realistic that a single adaptation node (or module) could cope with all kinds of usage environments [4]. As a consequence, different adaptation nodes distributed over the whole network could be employed, specifically for serving different access networks. Interoperability among these nodes can be guaranteed through standardized media and metadata formats. This requires that the metadata associated with the multimedia content needs to be transported to such adaptation nodes in order to steer the actual adaptation process there.

In [5], an architecture for dynamic and distributed multimedia content adaptation in streaming environments is proposed which introduces some of the key concepts described in this paper. This paper, however, concentrates on the details of transporting media and metadata between adaptation nodes within a heterogeneous network. Thus, this paper aims to provide answers for the unsolved issues in [5] focusing on metadata transport questions.

The remainder of this paper is organized as follows. Section II provides a qualitative evaluation of the possibilities how to transport the different types of metadata in conjunction with the actual multimedia content. In Section III, we propose a transport format for content-related metadata. Preliminary measurements and results can be found in Section IV and Section V concludes the paper.

II. EVALUATION OF MEDIA AND METADATA TRANSPORT MECHANISMS

A. Introduction

As described in [3] three different types of information are required by an adaptation node located within a network: (1) the description of the usage environment, e.g., in terms of terminal and network capabilities (among others), (2) the actual multimedia content, and (3) its associated metadata. DIA specifies two types of content-related metadata used within the adaptation engine as described in [3] and [5]. First, the generic Bitstream Syntax Description (gBSD) provides means for describing the syntax of a bitstream independent from its coding format. Second, the AdaptationQoS (AQoS) description specifies the relationship between usage environment, possible adaptation operations and resulting qualities of the content. Both types of metadata are used to adapt the multimedia content independent of its actual coding format.

The usage environment description is usually attached to the request for the content. In the sequel, however, we will concentrate on content-related metadata (gBSD and AdaptationQoS). In many use cases, streaming of media resources such as audio/video content is required, which is facilitated by the Real-time Transport Protocol (RTP). Due to its size it is unfeasible to transport the content-related metadata in one big chunk. Thus, it should be transported using RTP as

[1] Part of this work was supported by the European Commission in the context of the DANAE project (IST-1-507113); http://danae.rd.francetelecom.com

well, e.g., to exploit the synchronization mechanisms offered by RTP. However, there are still three possibilities how to transport these different metadata assets with the actual media content which are evaluated in the following:

- One combined stream containing media and metadata.
- One metadata stream and one media data stream.
- Multiple metadata streams (one for each type of metadata) and one media data stream.

B. Multiplexing in the Media Resources' Stream

Some RTP payload formats, such as the RTP payload format for transport of MPEG-4 Elementary Streams (RFC 3640) provide means for including arbitrary data, e.g., metadata, within the auxiliary header.

The *advantages* of this approach are that it is straight forward to implement (all the requisites are already specified) and that there is little processing and bandwidth overhead because there is only one stream to handle. Moreover there is no synchronization necessary between different streams, which reduces complexity.

However, this approach also has several *disadvantages*. The first one is based on the assumption that metadata is more valuable than the media data. Metadata, specifically the AdaptationQoS, usually describes many media access units (AUs) in a single metadata access unit (MAU) and therefore a large segment of the stream would be affected if such a MAU were lost. This issue raises the need for reliable transport mechanisms for MAUs. While re-transmission can be used to fulfill this requirement, it results in the need for large buffers, which increases the startup delay. A better solution would be to reserve enough bandwidth for the metadata stream in advance, which is not possible with the approach of a combined stream. Another – maybe the biggest – disadvantage is that the combined stream approach depends on a specific payload format (e.g., RFC 3640) which provides the auxiliary header section where the metadata can be transported. Other payload formats might not provide such an auxiliary section. That is, by following the combined stream approach one would create a solution which is limited to a specific type of resource. This conflicts with the need for interoperability. Moreover, while one saves processing overhead by having only one stream, there is some additional overhead due to the necessary (de)multiplexing of the media data and the metadata.

C. One Separate Medadata Stream

In this scenario, the different types of metadata are multiplexed into one metadata stream.

The *advantages* of this approach are that it allows the metadata stream to be treated differently from the media stream. This makes it possible to reserve bandwidth for the metadata stream in order to protect it from packet loss. Additionally, the transport of metadata is no longer bound to a specific media payload format.

The *disadvantages* of this approach are the additional processing overhead caused by the (de)multiplexing of the metadata, the additional bandwidth overhead due to the second RTP stream and the necessary synchronization between the two streams, which increases the complexity of the implementation.

Table 1: Advantages and disadvantages of different metadata transport mechanisms.

	One Combined Stream	One Metadata Stream	Multiple Metadata Streams
(De-)Multiplexing Efforts	High	Medium	**Low**
(De-)Packetizing Efforts	**Low**	Medium	High
Number of Streams	**1**	2	3+
Transport Overhead	**Low**	Medium	High
Processing Overhead	High	Medium	**Low**
Synchronization Efforts	**Low**	Medium	High
Interoperability Issues	Yes	**No**	**No**
Protection Flexibility	Low	Medium	**High**
Asynchronous Transport	No	Yes (limited)	**Yes**
Scalability	Yes (limited)	Yes (limited)	**Yes**

D. Multiple Metadata Streams

In this scenario, separate streams for each type of metadata are used, for example, one for the gBSD and one for the AdaptationQoS.

In addition to the *advantages* listed for the approach in Section C this mechanism offers the possibility of handling each kind of metadata by specialized adaptation nodes, e.g., an adaptation node with special hardware for XML processing[2], thus facilitating scalability. This is also possible for the other approaches, by de-multiplexing the stream(s) and then sending each type of metadata to the specialized adaptation nodes for processing. It would, however, introduce additional delay into the streaming chain. Another advantage lies in the possibility to send the AdaptationQoS and the gBSD slightly in advance (asynchronous transport) in order to be processed by the adaptation node before the media data arrives and is adapted. This results in lower startup delay as the adaptation node can more efficiently use its resources. The final advantage is that no metadata (de-)multiplexing is needed.

This last advantage of course comes at the price of the *disadvantage* of high bandwidth and packetizing overhead due to having multiple media and metadata streams. Moreover, synchronization of these three streams is needed and leads to additional complexity.

Table 1 summarizes the advantages and disadvantages of each of these possibilities. As a conclusion we will concentrate on the third option in the remainder of this paper.

III. TRANSPORT FORMATS AND STRATEGIES

In this section, we will first concentrate on the transport format and subsequently provide an example for it.

A. Transport Formats

In the following, we focus on investigating an RTP payload format for content-related metadata. First, the payload format is investigated and then the header fields which are used to signal information about the payload are discussed.

[2] XA35 XML Accelerator; http://www.datapower.com/products/xa35.html.

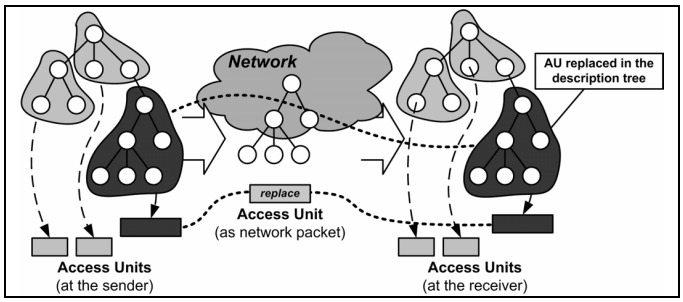

Figure 1: Streaming XML using BiM enables partial updates of the XML document (cf. highlighted nodes).

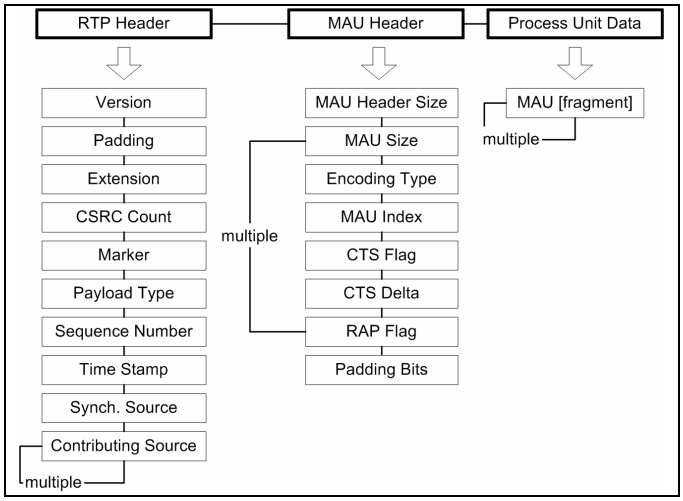

Figure 2: RTP packet with MAU payload.

1) Payload Format

The approach illustrated in [5] requires the fragmentation of the content-related metadata into independent XML fragments, called Process Units (PUs) for the gBSD and Adaptation Units (ADUs) for the AdaptationQoS. We will refer to both types as MAUs.

We have identified three options for the payload format of such MAUs. One can either transport them (1) in plain text, (2) compressed using a generic or an XML-aware compression algorithm, or (3) compressed with the MPEG-7 Binary Format for Metadata (BiM) [8] which allows streaming of XML-based data as depicted in Figure 1. While complete MAUs would be transmitted in the first two cases, BiM would signal a complete MAU only once and subsequently just the nodes which changed (see highlighted nodes and dotted lines in Figure 1), together with information on how and where to include them into the previously sent complete MAU. Please refer to Section B for an example of this approach. The transport encoding of the MAU is signaled using a payload specific header field. Independent from the encoding, each RTP packet may contain a fragment of an MAU, a complete MAU or many MAUs. In case of many MAUs, the MAU header indicates the length of each MAU as described in the next section.

2) Header Fields

Generally, the RTP header fields are used as defined in [10] and shown in Figure 2. The *Time Stamp* should be based on the same clock as the one which is used for the media content. Each MAU which describes a specific media segment carries the same *Time Stamp* as the first AU of that segment. If several MAUs describe a single AU, all MAUs carry the same *Time Stamp* as the AU.

```
<dia:DIA>
 <dia:DescriptionMetadata>
  <dia:ClassificationSchemeAlias alias="MV4"
    href="urn:mpeg:mpeg4:video
          :cs:syntacticalLabels"/>
 </dia:DescriptionMetadata>
 <dia:Description xsi:type="gBSDType"
         bs1:bitstreamURI="akiyo.mpg4">
  <gBSDUnit syntacticalLabel=":MV4:VO" start="0"
      length="18"/>
  <gBSDUnit syntacticalLabel=":MV4:I_VOP"
      start="18" length="4641" marker="Temporal-0"/>
  <gBSDUnit syntacticalLabel=":MV4:P_VOP"
      start="4659" length="98" marker="Temporal-1"/>
  <gBSDUnit syntacticalLabel=":MV4:B VOP"
      start="4757" length="16" marker="Temporal-2"/>
  <!-- ... and so on ... -->
 </dia:Description>
</dia:DIA>
```

Document 1: VES gBSD fragment.

For the MAU header we propose the following fields. The *MAU Header Size* indicates the size of the header. The *MAU Size* indicates the size of the associated MAU. The *Encoding Type* defines the encoding of the MAU. The *MAU Index* indicates the serial number of the associated MAU. The *CTS Flag* indicates if the composition time stamp of the MAU is available. If so, then the *CTS Delta* field contains the difference of the composition time stamp and the RTP time stamp. This can be used to transmit MAU packets in a different order than in which they are processed, e.g., in order to enable traffic smoothing. The *RAP Flag* indicates if the current packet is a random access point, i.e., if it contains a complete MAU. In case of BiM, this is the case whenever a complete MAU is transmitted. The RAPs of the MAU stream should be aligned with the RAPs of the media stream in order to provide common entry points for clients which wish to join the session.

B. Examples

A fragment of a gBSD describing an MPEG-4 Visual Elementary Stream (VES) [6] which comprises the video object header and the first three video object planes (VOPs), is shown in Document 1. It also includes a *marker* attribute for each VOP which indicates its suitability for temporal scaling. While dropping B-VOPs is generally a good idea, dropping I- or P-VOPs is more problematic since other VOPs depend on them.

Document 2 shows a single PU. Due to the requirement that a PU needs to be able to be processed independently from all other PUs, the PU includes all its ancestor nodes. It can also be seen that the *start* attribute is set to zero. The reason for this is that each PU describes a segment of the content which is adapted independently from the other resource segments. The adaptation engine has no knowledge of how many bytes of the resource have been adapted so far and therefore each resource segment (and its corresponding PU) is treated in the same way as a new resource.

```
<dia:DIA>
 <dia:DescriptionMetadata>
  <dia:ClassificationSchemeAlias alias="MV4"
      href="urn:mpeg:mpeg4:video
            :cs:syntacticalLabels"/>
 </dia:DescriptionMetadata>
 <dia:Description xsi:type="gBSDType"
      bs1:bitstreamURI="akiyo.mpg4">
  <gBSDUnit syntacticalLabel=":MV4:I_VOP" start="0"
      length="4641" marker="Temporal-0"/>
 </dia:Description>
</dia:DIA>
```

Document 2: VES gBSD Process Unit (MAU).

```
<FragmentUpdateUnit>
 <FUCommand>replaceNode</FUCommand>
 <FUContext>/dia:DIA/dia:Description/gBSDUnit
 </FUContext>
 <FUPayload>
  <gBSDUnit syntacticalLabel=":MV4:I_VOP" start="0"
      length="4641" marker="Temporal-0"/>
 </FUPayload>
</FragmentUpdateUnit>
```

Document 3: VES gBSD transport format (using BiM, textual representation).

As mentioned above, there are two options for the transport of the MAUs. One option is to transmit the complete MAU every time (cf. Document 2). The second option is the BiM approach which allows partial updates of the document, i.e., only the changes are transmitted to the adaptation node. BiM signals these changes using one or more so-called Fragment Update Units (FUUs) for each MAU. Document 3 shows a textual representation of such an FUU of the equivalent binary version for better readability. The *Fragment Update Command* (*FUCommand*) signals how the *Fragment Update Payload* (*FUPayload*) should be applied to the node referenced by the *Fragment Update Context* (*FUContext*). This allows for a very efficient signaling of MAUs, which we will evaluate in the following.

IV. MEASUREMENTS AND RESULTS

Compared to traditional server-centric adaptation, our proposal can offer three advantages. First, the adaptation is generic due to the usage of the content-related metadata. This allows the actual adaptation engine to be kept simple and efficient. Second, our approach also allows to quickly react to local bandwidth fluctuations, which may otherwise result in uncontrolled packet loss. Third, it reduces the bandwidth requirements in certain cases. In this section, we will take a look at this third characteristic.

We assume a multicast scenario, where a number of clients consume the same content, either through the adaptation node or directly from the server as described in [5]. Up to four clients are considered which all consume the same content but with different quality and bandwidth requirements. They may use different access networks, for instance. It is assumed that the first client wants optimum quality, the second client wants a quality corresponding to a bandwidth reduction of 25%, the third client wants a quality corresponding to a bandwidth reduction of 50%, and the fourth client wants a quality corresponding to a bandwidth reduction of 75%.

[3] XMLPPM 0.96; http://sourceforge.net/projects/xmlppm.

Table 2: MPEG-4 Visual Elementary Stream (VES) [6], MPEG-4 Bit Sliced Arithmetic Coding (BSAC) [7], and Embedded Zero Block Coding (EZBC) [9] sample media streams.

	Scalability Mode(s)	FPS	Characteristic
MPEG-4 VES	Temporal (Frame Dropping)	20	352x288 Pixel
MPEG-4 BSAC	Quality (48 Levels)	25	44kbps
EZBC	Quality, Spatial, Temporal	20	176x144 Pixel

Table 3: Sample media stream and metadata bandwidth requirements [kbps].

	Content	CRM plain text	CRM XMLPPM	CRM BiM
MPEG-4 VES	86,98	63,13	37,5	6,56
MPEG-4 BSAC	66,78	248,00	112,88	46,88
EZBC	1313,68	694,40	58,24	55,68

Table 4: Bandwidth requirements for traditional server-centric adaptation [kbps].

	1 Client	2 Clients	3 Clients	4 Clients
VES Server-Centric	86,98	152,22	195,71	217,46
BSAC Server-Centric	66,78	116,87	150,26	166,96
EZBC Server-Centric	1313,68	2298,94	2955,78	3284,2

Table 5: Bandwidth requirements for distributed adaptation [kbps].

	Plain Text	XMLPPM	BiM
VES including CRM	150,11	124,48	93,54
BSAC including CRM	314,78	179,66	113,66
EZBC including CRM	2008,08	1371,92	1369,36

Three types of content are used for our measurements which are listed in Table 2. Table 3 shows the bandwidth requirements (in kbps) of the sample data, including the content-related metadata (CRM) in plain text, with XML-aware XMLPPM[3] compression, and using the BiM approach (with zLib optimized codec for strings and binary context path encoding enabled).

In our measurements we will compare the bandwidth consumed in the server-centric approach (one stream for each client) and in the distributed approach. While there is one stream for each client in the server-centric approach, the distributed approach only requires one content stream and one content-related metadata stream between the server and the adaptation node. The adaptation node can then adapt the media in a generic way for any content quality which is asked for by the clients. The objective is to show for which amount of clients our proposal reduces the bandwidth requirements between the server and the adaptation node compared to server-centric adaptation. We will neglect the overhead of the transport mechanism (i.e., we are only looking at the payloads) and we will focus on the gBSD for the measurements.

Table 4 and Table 5 show a comparison of the bandwidth requirements (in kbps) for the server-centric scenario and the distributed scenario. Table 4 shows the server-centric scenario with one stream for each client and Table 5 shows the distributed scenario with one media and one metadata stream which is coded/compressed using the three different approaches which were introduced above.

The measurements on our test system show that with a single client, our approach always requires more bandwidth

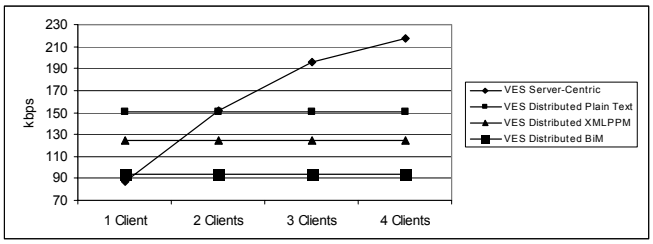

Figure 3: VES bandwidth requirements.

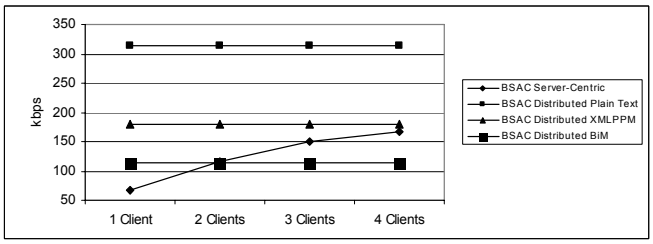

Figure 4: BSAC bandwidth requirements.

than server-centric adaptation due to the metadata overhead. For two or more clients, our proposal is more bandwidth efficient depending on the complexity of the content-related metadata which depends on the offered scalability modes.

VES offers only temporal adaptation for which our approach is more efficient than server-centric adaptation given two or more clients (cf. Figure 3). EZBC offers more diverse adaptation possibilities but due to the high bandwidth requirements of EZBC streams, our approach is more bandwidth efficient for two or more clients (cf. Figure 5). BSAC offers fine grained quality which means that the content-related metadata needs considerable bandwidth. Moreover audio streams need considerable less bandwidth than video streams. Here our approach is more efficient than server-centric adaptation when there are three or more clients with content-related metadata compressed using the BiM approach (cf. Figure 4).

Generally one can say that, in terms of size, the content-related metadata does not scale as well as media content. This results in considerable metadata overhead for media content that offers a wide range of scalability modes with low bandwidth requirements. For a BSAC stream with only 22 kbps, the content-related metadata would for example still be the same size as for the stream in our experimental setup.

One can also see from the measurements that the BiM codec performs best for compression of the content-related metadata. It also has the advantage of allowing to process the PUs in binary format, a possibility that could result in considerably less processing effort on the adaptation node.

Our approach offers benefits which are not available with traditional server-centric adaptation. Additionally it offers an increased bandwidth efficiency if several clients consume the same content.

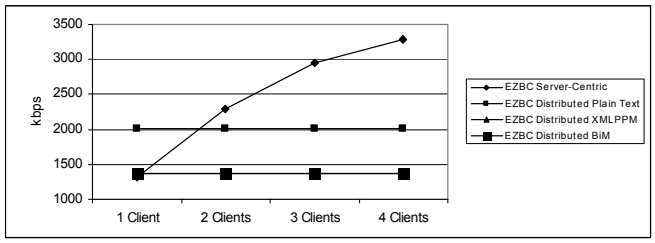

Figure 5: EZBC bandwidth requirements.

V. CONCLUSION

In this paper we presented a transport mechanism for content-related metadata enabling generic distributed content adaptation within the multimedia delivery chain. We have evaluated several approaches for metadata transport in conjunction with the actual media data it describes. Finally, we provided first results demonstrating the benefits of our approach.

Future work will result in a complete implementation of our distributed adaptation approach in the context of the DANAE project. It will also be investigated how our approach can be beneficial in unicast scenarios, e.g., with the support of caching mechanisms. Further research will also evaluate our approach in load balancing scenarios where different types of adaptations are conducted on distributed, specialized adaptation nodes.

REFERENCES

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

[2] A. Vetro and C. Timmerer, “Digital Item Adaptation: Overview of Standardization and Research Activities”, *IEEE Trans. on Multimedia*, vol. 7, no. 3, June 2005.

[3] C. Timmerer and H. Hellwagner, “Interoperable Adaptive Multimedia Communication”, *IEEE Multimedia Magazine*, vol. 12, no. 1, Jan.-Mar. 2005, pp. 74-79.

[4] K. Leopold, D. Jannach, H. Hellwagner, “A Knowledge and Component Based Multimedia Adaptation Framework”, *Proc. of the IEEE 6th Int’l. Symposium on Multimedia Software Engineering (MSE)*, Florida, USA, Dec. 2004, pp. 10-17.

[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”, accepted for Special Session on Adaptive Wireless Video Streaming at *IEEE Int’l. Conf. on Image Processing 2005*, Genova, Italy, Sep. 2005.

[6] T. Ebrahimi, C. Horne, "MPEG 4 Natural Video Coding - An Overview," *Image Communication Journal*, vol. 15, no. 4-5, Jan. 2000, pp. 365-385.

[7] H. Purnhagen, "An Overview of MPEG-4 Audio Version 2", *AES 17th International Conference on High-Quality Audio Coding*, Firenze, Italy, Sep. 1999.

[8] 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.

[9] S.-T. Hsiang and J. W. Woods, “Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling”, *MPEG-4 Workshop and Exhibition at ISCAS 2000*, Geneva, Switzerland, May 2000.

[10] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RFC 3550 – RTP: A Transport Protocol for Real-Time Applications, July 2003.