NinSuna: a fully integrated platform for format-independent multimedia content adaptation and delivery using semantic web technologies

Davy Van Deursen, Wim Van Lancker, Wesley De Neve, Tom Paridaens, Erik Mannens, and Rik Van de Walle

In: Multimedia Tools and Applications – Special Issue on Data Semantics for Multimedia Systems, Volume 46, Numbers 2-3, January, 2010

http://www.springerlink.com/content/461380502m756877/

D. Van Deursen, W. Van Lancker, W. De Neve, T. Paridaens, E. Mannens, and R. Van de Walle (2010). NinSuna: a fully integrated platform for format-independent multimedia content adaptation and delivery using semantic web technologies. *Multimedia Tools and Applications – Special Issue on Data Semantics for Multimedia Systems* 46(2-3) pp. 371-398. 10.1007/s11042-009-0354-0

NinSuna: a Fully Integrated Platform for Format-independent Multimedia Content Adaptation and Delivery using Semantic Web Technologies

Davy Van Deursen, Wim Van Lancker, Wesley De Neve[*], Tom Paridaens, Erik Mannens, Rik Van de Walle

Ghent University – IBBT
Departement of Electronics and Information Systems – Multimedia Lab
Gaston Crommenlaan 8, bus 201, 9050 Ledeberg-Ghent, Belgium
{davy.vandeursen,wim.vanlancker}@elis.ugent.be, wesley.deneve@kaist.ac.kr,
{tom.paridaens,erik.mannens,rik.vandewalle}@elis.ugent.be

Abstract

The current multimedia landscape is characterized by a significant heterogeneity in terms of coding and delivery formats, usage environments, and user preferences. The main contribution of this paper is a discussion of the design and functioning of a fully integrated platform for multimedia adaptation and delivery, called NinSuna. This platform is able to efficiently deal with the aforementioned heterogeneity in the present-day multimedia ecosystem, thanks to the use of format-agnostic adaptation engines (i.e., engines independent of the underlying coding format) and format-agnostic packaging engines (i.e., engines independent of the underlying delivery format). Moreover, NinSuna also provides a seamless integration between metadata standards and adaptation processes. Both our format-independent adaptation and packaging techniques rely on a model for multimedia bitstreams, describing the structural, semantic, and scalability properties of these multimedia streams. News sequences were used as a test case for our platform, enabling the user to select news fragments matching his/her specific interests and usage environment characteristics.

Keywords: BSDL, Format-independent, Multimedia adaptation, Multimedia delivery, Multimedia model, Semantic Web, XML

[*]At the time of writing, Wesley De Neve was also with the Image and Video Systems Lab of the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea.

1 Introduction

Recent years have witnessed an increasing heterogeneity in the multimedia landscape on different fronts. First, there is a growing diversity in end-user devices that are able to consume multimedia content. In particular, these devices may vary in terms of screen size, processing power, and battery life. Next, network technologies, used to transport the multimedia content to the end-user, may differ in terms of bandwidth, jitter, and error robustness. Furthermore, the number of multimedia coding standards has grown significantly over the last few years, especially with the introduction of new coding formats such as H.264/AVC [29], Scalable Video Coding (SVC, [38]), AAC [28], and JPEG XR [40]. At the same time, older standards such as H.262/MPEG-2 Video [27], MPEG-1 Audio [26], and JPEG2000 [11] are still present. Next to coding formats, there also exists a wide variety of delivery formats, i.e., formats encapsulating encoded multimedia streams (e.g., the MP4 file format [22] or the Real-time Transport Protocol (RTP, [1])). Finally, end-users with specific preferences often want to obtain a personalized version of multimedia content (e.g., an end-user only requesting scenes satisfying his/her interests). Therefore, metadata have an increasingly important role in multimedia search, adaptation, and delivery processes because they enable the effective organization, access, and interpretation of multimedia content.

The major contribution of this paper is the design and functioning of a fully integrated platform for multimedia adaptation and delivery. This platform is able to efficiently deal with the heterogeneity in the current and future multimedia landscape in terms of coding and delivery formats, usage environments, and user preferences. To satisfy these requirements and to obtain Universal Multimedia Access (UMA, [43]), our platform relies on format-agnostic adaptation engines (i.e., engines independent of the underlying coding format) and format-agnostic packaging engines (i.e., engines independent of the underlying delivery format). Moreover, the proposed platform enables a seamless integration between metadata standards and adaptation processes.

Format-agnostic content adaptation techniques have already been extensively described in the scientific literature [15, 34]. These techniques[1] are usually based on automatically generated XML-based descriptions of the high-level structure of multimedia resources [8]. The XML descriptions, further denoted as Bitstream Syntax Descriptions (BSDs), typically contain high-level syntax elements and pointers to data ranges in the original bitstream. BSDs are used to realize simple, high-level adaptation operations on compressed bitstreams. Examples are the removal of particular data blocks or the modification of the value of certain syntax elements. The actual adaptation takes place in the XML domain by transforming the BSDs using XML filters. A transformed BSD and its corresponding (original) bitstream are then used to create an adapted bitstream. The Bitstream Syntax Description Language (BSDL, [9]) is one such example of a standardized, format-independent content adaptation tool. BSDL comes with two standardized, format-independent parsers: a BintoBSD parser that is responsible for producing BSDs and a BSDtoBin parser that is used for generating adapted bitstreams. Both parsers are steered by a Bitstream Syntax Schema (BS Schema), which contains a description of the high-level structure and syntax elements of a particular coding format [20]. Other examples of format-independent content adaptation techniques are MPEG-21 generic Bitstream Syntax Schema (gBS Schema, [23]) and XFlavor [19].

Because format-agnostic content adaptation techniques try to abstract the adaptation process, some restrictions regarding the adaptation possibilities need to be taken into account. The use of format-independent content adaptation for instance implies that only high-level bitstream structures can be removed and that only high-level syntax elements can be modified. In other words, the compressed media bitstreams need to be encoded in such a way that it is possible to perform the adaptations without the need of a complete or partial recode process (which is computationally complex, compared to bitstream extraction). Therefore, typical target adaptation operations are the exploitation of scalability in multimedia bitstreams and high-level semantic adaptations. Exploitation of scalability is performed to adapt media bitstreams in order to meet the terminal and network characteristics of the end-user. Lowering the frame rate or the spatial resolution are examples of such adaptation operations. High-level semantic adaptations are adaptations based on semantic information about the multimedia content. Examples of semantic adaptations are the

[1]In the remainder of this paper, 'XML-driven content adaptation' is used as a collective term to refer to these techniques.

selection of a Region Of Interest (ROI) or the selection of specific temporal segments that are of interest to the user.

In this paper, we present the design and functioning of NinSuna[2], which is a fully integrated and format-independent platform for the purpose of multimedia adaptation and delivery. Its basic design is inspired by the principles of XML-driven content adaptation techniques, while its final design consists of a hybrid architecture using both XML and Semantic Web technologies such as the Resource Description Framework (RDF, [30]), the Web Ontology Language (OWL, [31]), and the SPARQL Protocol And RDF Query Language (SPARQL, [36]). Furthermore, a tight coupling exists between NinSuna's design and a model for describing the structural, semantic, and scalability properties of multimedia streams. This model, implemented using OWL, provides support for a seamless integration of adaptation operations and semantic metadata. Furthermore, it allows format-independent packaging and delivery of multimedia content.

2 Motivation

As discussed in the introduction, current format-independent content adaptation techniques are based on automatically generated XML descriptions (BSDs), i.e. *structural metadata*. Despite its format-independent nature, XML-driven content adaptation has a number of disadvantages. The XML filters are dependent on the structure of the metadata and underlying coding formats. Furthermore, due to interoperability problems between XML-based metadata standards [14, 10], integration with *semantic metadata* occurs in an ad-hoc manner. Mappings between different XML-based metadata formats need to be implemented in different XML filters. Hence, creators of these XML filters cannot think in terms of high-level adaptation operations but have to be aware of the underlying coding and metadata formats. For example, expressing the semantic adaptation operation *"select sport fragments"* in a scenario where two different content metadata standards are used (e.g., annotations are provided in both MPEG-7 and TV-Anytime), requires the development of two different XML filters (i.e., one that can interpret MPEG-7 and one that can interpret TV-Anytime).

The interoperability problems between XML-based metadata standards are due to the fact that XML Schema just describes grammars. There is no way to recognize a semantic unit from a particular domain because XML aims at document structure and imposes no common interpretation of the data contained in the document. For instance, taking into account different metadata standards, the same tags can have a different meaning, while tags with the same meaning can occur in different structures. Therefore, as will be illustrated in Section 3, Semantic Web technologies such as RDF and OWL can be used to enhance the interoperability among different metadata standards for multimedia content, thanks to the natural representation of objects and relationships [7, 14, 10].

A logical step after the adaptation of multimedia content is multimedia delivery. Multimedia content is usually not delivered as elementary bitstreams but packed in a particular delivery format. Today, a significant number of delivery or packaging formats exists; examples are MPEG-4 Part 14 (MP4 file format, [22]), Material eXchange Format (MXF, [39]), and Real-time Transport Protocol (RTP, [1]). As illustrated in Figure 1, we have to deal with different coding formats on the one hand, and different delivery formats on the other hand. Our goal is to develop a format-independent multimedia packager, i.e., a generic software module that is independent of the incoming coding format and the outgoing delivery format. An additional challenge in the context of this paper is the coupling of format-independent multimedia adaptation with format-independent multimedia packaging.

3 Format-independent Multimedia Content Adaptation

In this section, we present a new multimedia content adaptation framework inspired by the principles of XML-driven content adaptation techniques. Its design is based on Semantic Web technologies and a model for multimedia streams covering the structural, semantic, and scalability

[2]NinSuna is short for "The NinSuna INtelligent Search framework for UNiversal multimedia Access".

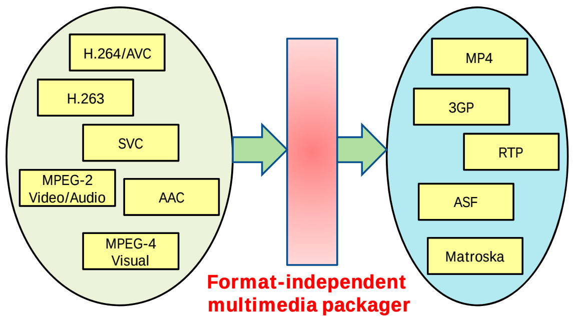


Figure 1: Obtaining a format-independent multimedia packager.

properties of these multimedia streams. In Section 4, our adaptation framework will be extended in order to support the format-independent packaging of adapted multimedia content.

3.1 Model for Multimedia Bitstreams

The model for multimedia bitstreams provides support for a seamless integration of adaptation operations and semantic metadata. As such, it enables the definition of adaptation operations on a higher level (i.e., based on the model), on the condition that current and future coding formats can be mapped to this model. We have avoided the use of XML Schema to describe our model because the use of XML as underlying technology causes interoperability problems between different metadata standards, as discussed in Section 2. In contrast to XML, Semantic Web technologies such as RDF and OWL enhance the interoperability among metadata standards for multimedia content. Therefore, our model for media bitstreams is implemented as an OWL DL ontology. The instances of the model (i.e., the structural metadata or BSDs) are expressed in RDF. The transformation of the structural metadata is implemented by using SPARQL queries, which are independent of the coding format. A visualization of an excerpt of our model is given in Figure 2.

The structural metadata part of the model (1) describes information regarding the high-level structure of a compressed *MultimediaBitstream*. Such a *MultimediaBitstream* points to the physical location of the multimedia bitstream by means of the *bitstreamSource* property. Further, it also provides the MIME type of the underlying coding format[3] by means of the *format* property.

A *MultimediaBitstream* points to a list of *RandomAccessUnits* by means of the *hasStructure* property. Random access refers to the ability of the decoder to start decoding at a point in a compressed multimedia bitstream other than at the beginning and to recover an exact representation of the decoded bitstream [18]. Note that *RandomAccessUnits* cannot overlap with each other.

Each random access unit points to a list of *DataBlocks* by means of the *hasStructure* property. Note that this property is transitive in order to express that *DataBlocks* are always contained in *MultimediaBitstreams* (even if they are located within a *RandomAccessUnit*). A *DataBlock* points to a particular segment of the compressed multimedia bitstream by means of a bitstream position and a bitstream length in terms of bits[4]. *DataBlocks* that must be available in every resulting multimedia bitstream (e.g., Sequence or Picture Parameter Sets in H.264/AVC) are data blocks that do not belong to any random access unit. More specifically, the multimedia bitstream points directly to these data blocks by means of the *hasStructure* property. Note that a DataBlock does not necessarily represent a well-defined structure in a coding format (such as a frame or slice); it just represents a single byte range pointer. Further, *DataBlocks* cannot overlap with each other. Additionally, a data block can contain ScalabilityInformation (3), indicating to which scalability layers the data block belongs (in the assumption that the underlying bitstream is a scalable bitstream [33]).

The semantic metadata part of the model (2) contains a content description of the multimedia bitstream (i.e., *AnnotatedMultimedia*). *MultimediaBitstreams* are linked to an *AnnotatedMulti-*

[3]The *mime-type* class is defined in the Core Ontology for MultiMedia (COMM, [10]).

[4]To express the unit of measure, we use concepts of the Suggested Upper Merged Ontology (SUMO, [2]). Note that we use the OWL DL implementation of SUMO, which is available at `http://www.stuarthendren.net/sumodl`.

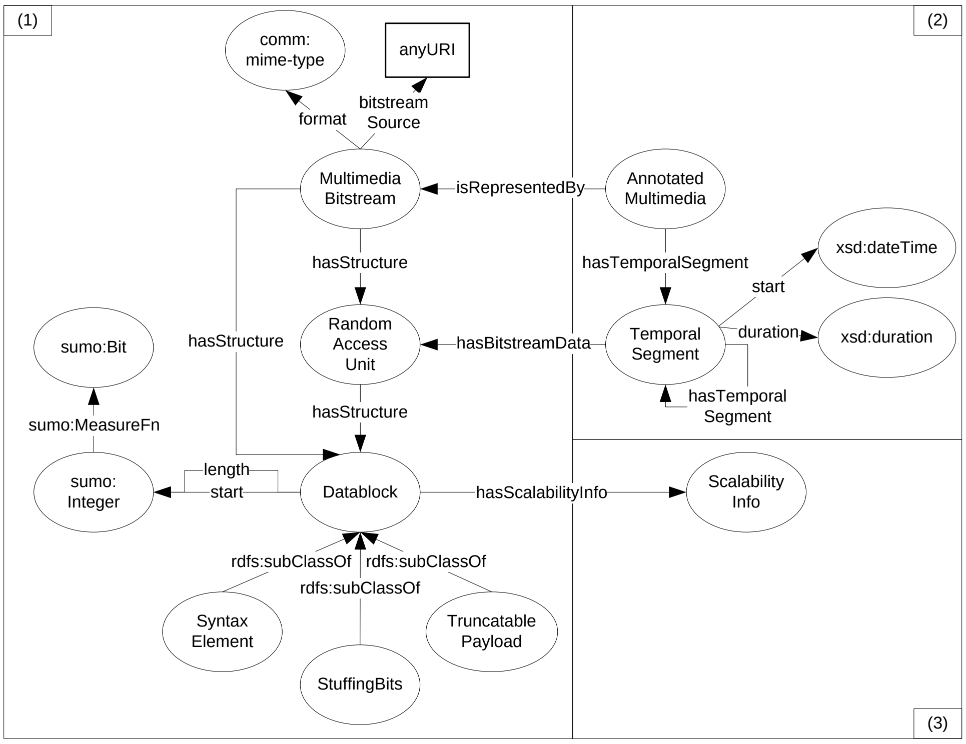


Figure 2: Excerpt of the model for multimedia bitstreams. Ellipses and arrows represent OWL classes and properties respectively.

media instance by means of the *isRepresentedBy* property. *AnnotatedMultimedia* points to a list of *TemporalSegment*s by means of the *hasTemporalSegment* property. *TemporalSegment*s can be decomposed further into other *TemporalSegment*s by means of the *hasTemporalSegment* property. Note that different *TemporalSegment*s can overlap with each other. Each *TemporalSegment* points to a specific segment of the multimedia content by means of timestamps (i.e., *start* and *duration*). The connection between the semantic and structural metadata is realized by linking temporal segments (i.e., timestamps) to random access units (i.e., bits) by means of the *hasBitstreamData* property. Furthermore, already existing (domain-specific) ontologies can be linked to the semantic part of the model to create detailed semantic descriptions of the multimedia content.

Support for existing coding formats is provided by mapping them to the structural part of the model. When coding formats are mapped to the model and (domain-specific) ontologies are linked, instances of the model can be generated, resulting in a collection of RDF triples describing a particular multimedia bitstream. Several possibilities exist to generate metadata compatible with the model. For instance, the semantic metadata could be obtained using feature extraction algorithms or by (manual) annotation. The resulting semantic metadata will consist of instances of *AnnotatedMultimedia* and *TemporalSegment*s. The structural metadata could be generated during the encoding of a multimedia bitstream or by using generic software similar to the BSDL approach. As a result, a *MultimediaBitstream* instance is created accompanied by instances of *RandomAccessUnit*s, *DataBlock*s, etc. Note that the structural metadata generation process can take the semantic metadata as input in order to connect the structural metadata with the semantic metadata (i.e., to link the bits of the encoded bitstream to the timestamps available in the semantic metadata). As previously discussed in this section, this is realized by linking *RandomAccessUnit*s to *TemporalSegment*s by using the *hasBitstreamData* property.

3.2 RDF-driven Content Adaptation

RDF-driven content adaptation is a new technique for multimedia content adaptation in a format-independent way. On the one hand, a model for multimedia bitstreams covering the structural,

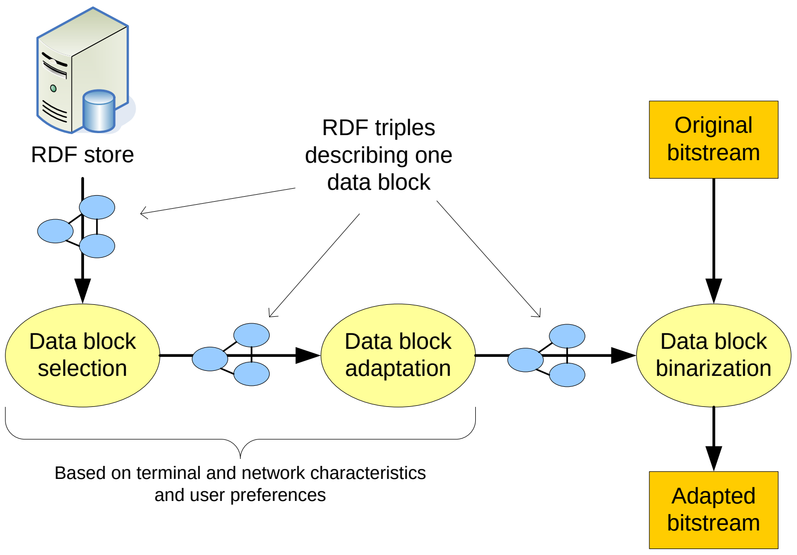


Figure 3: The general workflow of RDF-driven content adaptation.

```
PREFIX mmo: <multimedia_model.owl#>
CONSTRUCT {
  # triples to describe a datablock:
  ?db rdf:type mmo:DataBlock.
  # ...
}
WHERE {
  # select multimedia content based on a keyword:
  ?annoMM rdf:type mmo:AnnotatedMultimedia.
  ?annoMM mmo:hasTemporalSegment ?segment.
  ?segment dc:creator 'John Smith'.

  # select the corresponding datablocks:
  ?annoMM mmo:isRepresentedBy ?bitstream.
  ?bitstream rdf:type mmo:MultimediaBitstream.
  ?bitstream mmo:format 'video/H264'.
  ?bitstream mmo:hasStructure ?rau.
  ?segment mmo:hasBitstreamData ?rau.
  ?rau mmo:hasStructure ?db.
  ?db rdf:type mmo:DataBlock.
  # ...
}
```

Listing 1: SPARQL query selecting data blocks based on user preferences.

semantic, and scalability properties (discussed in Section 3.1) is used to abstract the adaptation process. On the other hand, Semantic Web technologies are used to represent the metadata for multimedia content and to support the adaptation process.

The general workflow to create an adapted version of a multimedia bitstream using RDF-driven content adaptation is depicted in Figure 3. For a particular multimedia bitstream, we assume that all RDF triples compliant with the model are present in an RDF store. The adaptation process consists of three steps: selection, adaptation, and binarization of data blocks. RDF graphs describing data blocks are queried during the data block selection step. These data blocks, corresponding to the requested multimedia content, are selected using a SPARQL query. An example of such a query is shown in Listing 1. Evaluating this query results in the construction of a list of data blocks corresponding to the requested content. Hence, RDF graphs describing these data blocks are selected, constructed, and further processed by the adaptation engine.

The primary goal of the data block transformation step is to make changes inside the selected RDF graphs describing a data block. For instance, the value of a *SyntaxElement* can be changed or the length of a *TruncatablePayload* can be shortened. A second use case for data block transformation is the support for dynamic adaptations, this is, when the multimedia content is delivered during varying usage environment conditions. That way, each time an adaptation property changes, the initialization and evaluation of a new query can be avoided.

The binarization step takes as input the selected and transformed data blocks, together with the original multimedia content. The adapted bitstream is created by copying parts of the original bitstream based on the bit positions of the selected data blocks.

Note that the SPARQL query shown in Listing 1 expresses the semantic adaptation operation *"select data blocks corresponding to temporal segments created by John Smith"*. To show why it is beneficial to make use of RDF instead of XML, we illustrate the selection of data blocks from multimedia bitstreams that were annotated using two different metadata formats (i.e., MPEG-7 [21] and Dublin Core [16]). Suppose we only want to extract fragments created by a specific person. When making use of XML-driven content adaptation and when relying on XML-based metadata, we need to take into account the structure and syntax of both MPEG-7 and Dublin Core descriptions. More specifically, we need to match the MPEG-7 creator tag (located in mp7:CreationInformation/mp7:Creation/mp7:Creator) and the Dublin Core creator tag (i.e., dc:creator). However, when the metadata is expressed in terms of RDF, we can state that the properties *mp7:Creator* and *dc:creator* are equal by using the *owl:equivalentProperty* property. Therefore, the SPARQL query shown in Listing 1 will select data blocks that are annotated by both MPEG-7 and Dublin Core metadata. Hence, using Semantic Web technologies to represent metadata enhances the interoperability between different metadata standards and allows us to express semantic metadata operations independent of the different underlying semantic metadata standards such as MPEG-7 and Dublin Core.

4 Format-independent Multimedia Content Packaging

Encapsulating multimedia content in a particular delivery format typically consists of two main processes: fragmentation and packetization. The fragmentation process divides the input multimedia bitstream into portions, each representing one fragment. A simple example of a fragment in the context of video streams is a single frame. The packetization process selects (and possibly aggregates) the obtained fragments and maps them to the output delivery format. This mapping includes the assignment of timestamps and the addition of syntactical structures such as packet headers.

In this section, we present a new method for format-independent multimedia content packaging. We use MPEG-B BSDL to abstract the packed multimedia bitstream and to enable the use of format-agnostic software modules. It is based on an extension of our model for multimedia bitstreams, as previously outlined in Section 3.1. Furthermore, a seamless integration is obtained between multimedia content adaptation and packaging in a format-independent way.

4.1 Extension of the Model for Multimedia Bitstreams

In Figure 4, an overview is given of the extensions added to the model for multimedia bitstreams in order to support format-independent packaging. Two categories can be distinguished: support for timestamps and support for delivery parameters.

A *timestamp* property is added to the *DataBlock* class, which represents a number related to the display time of the data block. In order to actually calculate the display time, the *timestampRate* property is added to the *MultimediaBitstream* class. The latter contains a number indicating the amount of timestamps that are contained in one second. Note that a data block does not necessarily correspond to a fragment (i.e., the outcome of the fragmentation process as described above). Multiple data blocks can make one fragment (e.g., data blocks representing H.264/AVC slices can be grouped in one fragment). However, one data block cannot be split up into several fragments.

Our model represents both the presentation and decoding timestamps. The presentation timestamp is explicitly defined by means of the *timestamp* property. The decoding timestamp is implicitly available through the order of occurrence of the data blocks (i.e., this order corresponds to the decoding order). In order to determine the decoding timestamp of an access unit (i.e., one or more data blocks having the same value for their timestamp property), we can apply the following

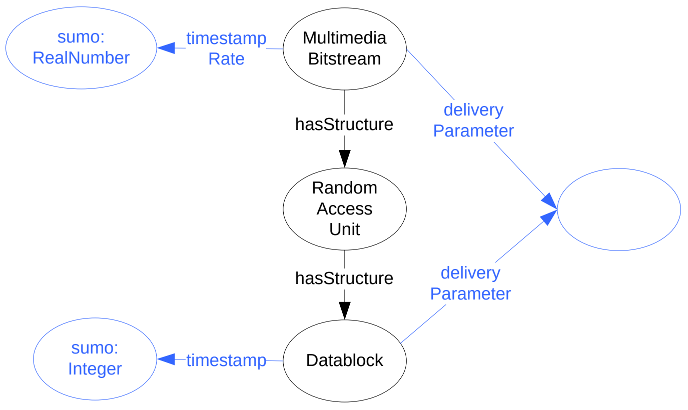


Figure 4: Extending the model for multimedia bitstreams to support format-independent multimedia packaging.

```
<#samplingFrequency> rdfs:subPropertyOf mmo:deliveryParameter .
<#audioMMBitstream> <#samplingFrequency> [
            a sumo:Integer;
            sumo:RealNumberFn "48000"^^<http://www.w3.org/2001/XMLSchema#integer>;
            sumo:MeasureFn sumo:Hertz ] .
```

Listing 2: RDF triples (in N3) expressing a delivery parameter corresponding to a sampling frequency equal to 48000 Hz.

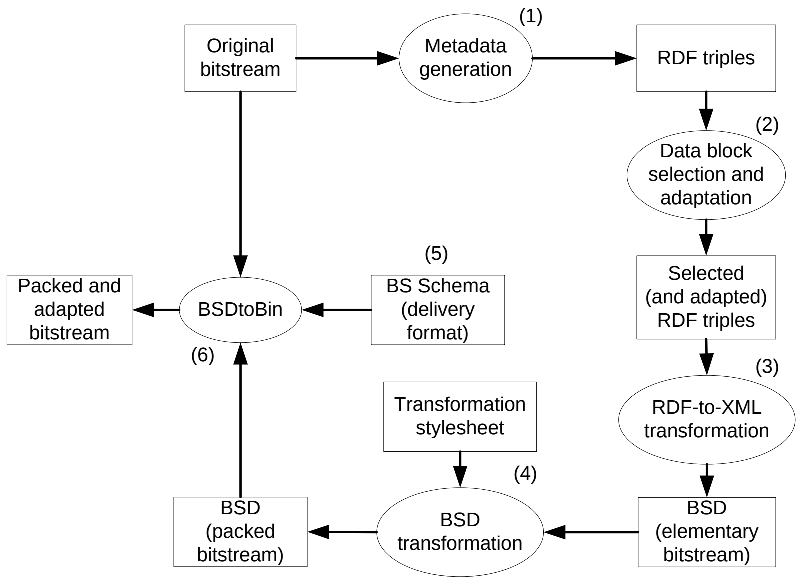
algorithm: the decoding time delta between two access units is equal to $1/timestampRate$. Hence, the decoding timestamps can be determined by combining the decoding order with these decoding time deltas.

Next, in order to assist in the fragmentation and packetization process, it is possible to define delivery parameters by means of the *deliveryParameter* property. This can be done at the level of a multimedia bitstream, as well as at the level of a data block. An example of a delivery parameter added at the level of a multimedia bitstream is the sampling frequency when the underlying coding format is AAC. The sampling frequency is a delivery parameter that is needed by the packetization process of AAC streams, since the value of this parameter may be needed in the headers of a particular delivery format. The *deliveryParameter* property serves as a superproperty for format-specific delivery parameters. For example, the property *samplingFrequency* can be defined as a subproperty of *deliveryParameter*. Listing 2 shows the RDF triples (in N3) expressing a delivery parameter corresponding to a sampling frequency equal to 48000 Hz.

4.2 Coupling RDF-driven Content Adaptation with Multimedia Packaging

The two extensions of our model for multimedia bitstreams enable the creation of metadata (compliant with the model) that can assist in the format-independent adaptation and packaging of multimedia bitstreams. The general workflow of RDF-driven content adaptation and packaging is depicted in Figure 5. Explanatory notes are given below.

(1) *Metadata generation*: multimedia bitstreams that need to be adapted and packaged in our framework have to be equipped with metadata compliant with our (extended) multimedia model. As discussed in Section 3.1, the metadata generation can occur during the encoding process or by means of a (generic or format-specific) software module. Note that such a generic solution could rely on the principles of techniques such as BSDL or XFlavor, where structural metadata is generated based on a description of the high-level structures and syntax elements of a particular coding format. The result of the metadata generation process is a collection of RDF triples compliant with the model for multimedia bitstreams.



(1)
Original bitstream
Metadata generation
RDF triples
(2)
Data block selection and adaptation
Selected (and adapted) RDF triples
(3)
RDF-to-XML transformation
BSD (elementary bitstream)
(4)
Transformation stylesheet
BSD transformation
BSD (packed bitstream)
(5)
BS Schema (delivery format)
(6)
BSDtoBin
Packed and adapted bitstream

Figure 5: The general workflow of RDF-driven content adaptation and packaging.

(2) *Data block selection and adaptation*: as discussed in Section 3.2, adaptation of multimedia bitstreams is performed by selecting the proper data blocks and by possibly adapting the selected data blocks.

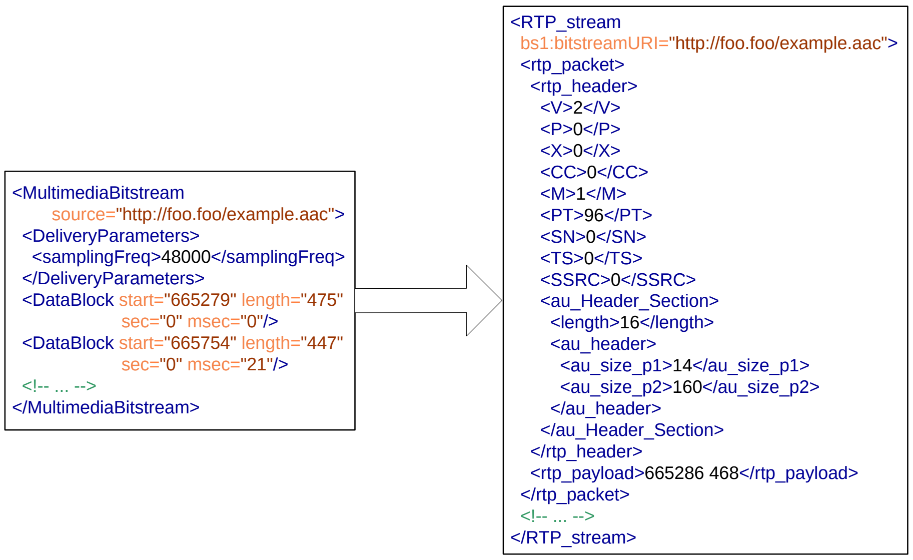
(3) *RDF-to-XML transformation*: instead of creating an adapted, elementary multimedia bitstream based on the selected (and adapted) data blocks, we perform a simple RDF-to-XML transformation. The result of this transformation is a BSD (see Section 1) which can be used to create a packaged version of the adapted multimedia bitstream. An example of such a BSD is shown on the left-hand side of Figure 6. The classes and properties defined in our model, needed for the packaging process, are mapped to XML elements and attributes respectively. Note that the timestamps are represented in terms of seconds and milliseconds.

(4) *BSD transformation*: the actual packaging process starts with the transformation of the BSD representing the adapted, elementary multimedia bitstream. The resulting BSD represents an adapted and packaged multimedia bitstream. Figure 6 illustrates the BSD transformation for the RTP packaging of an AAC bitstream. The obtained BSD is compliant with MPEG-B BSDL, which implies that the BSDL framework can be used for further processing. The BSD transformation can be implemented using XSLT or STX, which enables the use of a format-independent transformation engine. However, it is important to note that the transformation stylesheets are not only dependent on the target delivery format, but also on the incoming coding format since each coding format requires a different packaging (i.e., fragmentation and packetization) strategy.

(5) *BS Schema creation*: as discussed in Section 1, a BS Schema describes the high-level structures and syntax elements of a particular format. In this case, a BS Schema for the target delivery format needs to be created. The BSD obtained in the previous step needs to be compliant with this BS Schema.

(6) *Adapted and packed bitstream generation*: finally, an adapted and packaged multimedia bitstream can be created using BSDL's format-independent BSDtoBin parser, based on the BSD representing the adapted and packaged multimedia bitstream, the BS Schema for the target delivery format, and the original multimedia bitstream.

As discussed above, packaging multimedia bitstreams typically consists of two main processes: fragmentation and packetization. It is not trivial to see where these two processes actually occur in the above discussed workflow for RDF-driven content adaptation and packaging. Fragmentation is



```
<MultimediaBitstream
      source="http://foo.foo/example.aac">
  <DeliveryParameters>
    <samplingFreq>48000</samplingFreq>
  </DeliveryParameters>
  <DataBlock start="665279" length="475"
              sec="0" msec="0"/>
  <DataBlock start="665754" length="447"
              sec="0" msec="21"/>
  <!-- ... -->
</MultimediaBitstream>
```

```
<RTP_stream
  bs1:bitstreamURI="http://foo.foo/example.aac">
  <rtp_packet>
    <rtp_header>
      <V>2</V>
      <P>0</P>
      <X>0</X>
      <CC>0</CC>
      <M>1</M>
      <PT>96</PT>
      <SN>0</SN>
      <TS>0</TS>
      <SSRC>0</SSRC>
      <au_Header_Section>
        <length>16</length>
        <au_header>
          <au_size_p1>14</au_size_p1>
          <au_size_p2>160</au_size_p2>
        </au_header>
      </au_Header_Section>
    </rtp_header>
    <rtp_payload>665286 468</rtp_payload>
  </rtp_packet>
  <!-- ... -->
</RTP_stream>
```

Figure 6: BSD-driven RTP packaging of AAC multimedia bitstreams.

realized during the BSD transformation process, where the data blocks are mapped to fragments. Packetization is spread across multiple steps. One aspect is the assignment of timestamps to fragments. During the metadata generation step, the data blocks are labeled with initial timestamps (i.e., timestamps of the original multimedia bitstreams). However, since the adaptation process can cause gaps in the initial timestamps (e.g., a particular scene is deleted during the adaptation), these timestamps need to be recalculated (i.e., the gaps need to be detected and corrected). The latter is performed during the RDF-to-XML transformation. The packetization process also includes the addition of syntactical structures such as packet headers. This is done during the BSD transformation.

The choice to go back from RDF to XML during the packaging process can be justified as follows. We introduced Semantic Web technologies such as RDF to enhance the interoperability between different metadata standards. However, the latter is mainly an adaptation issue. More specifically, semantic adaptations such as scene selection based on semantic metadata are suffering from these interoperability problems (in case the adaptation is performed in the XML domain). In order to obtain packaging of multimedia bitstreams in a format-independent way, a description of headers and syntax elements of the target delivery format is needed, together with pointers to data segments in the original bitstream, which is exactly what already existing technologies such as MPEG-B BSDL support.

5 The NinSuna Platform

In this section, we introduce NinSuna[5], which is a fully integrated platform for multimedia adaptation and delivery in heterogeneous usage environments, relying on both XML and Semantic Web technologies for the implementation of format-independent adaptation and packaging engines. Furthermore, it aims at being deployable in streaming environments. Its multimedia content adaptation and packaging techniques rely on our model for multimedia bitstreams. Note that these adaptation and packaging techniques were previously discussed in Section 3 and Section 4 respectively. Next to the delivery of multimedia content, NinSuna also provides support for uploading content with corresponding metadata.

[5]A website containing information regarding the NinSuna platform and an online demo is available on `http://multimedialab.elis.ugent.be/NinSuna`.

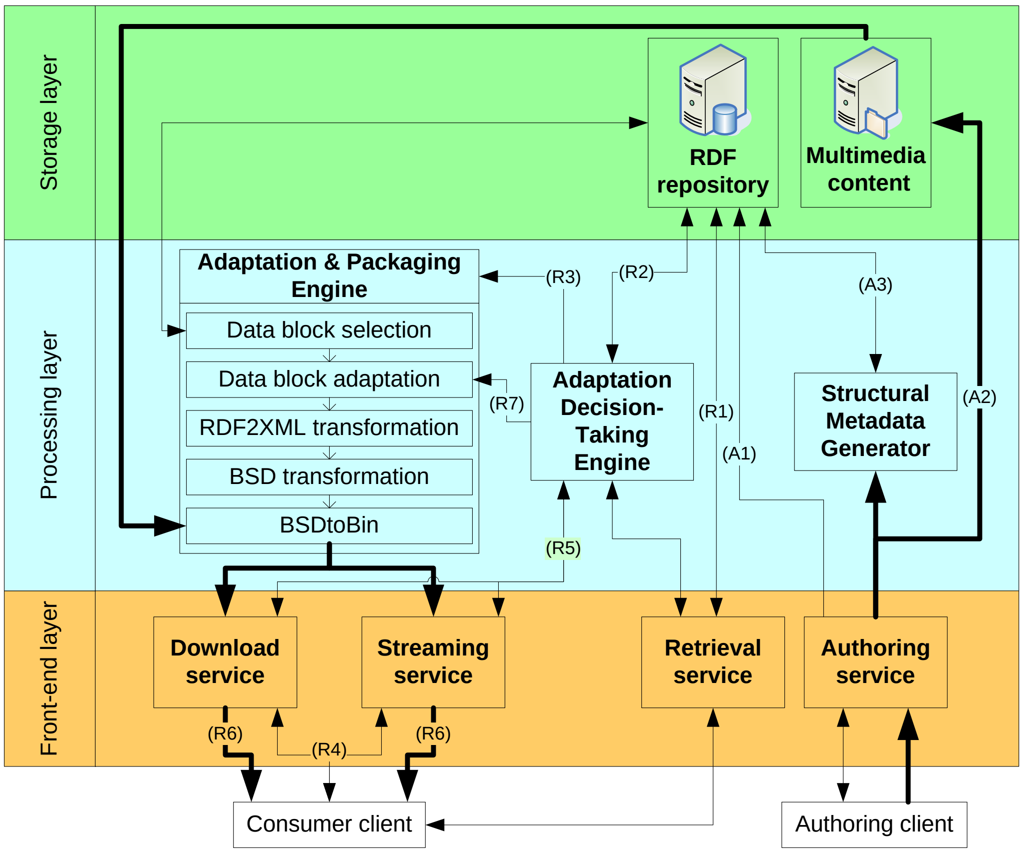


Figure 7: Architecture of the NinSuna platform.

5.1 Architecture

The NinSuna architecture is shown in Figure 7. Three layers can be distinguished: the storage layer, the processing layer, and the front-end layer. The storage layer consists of a multimedia content server, containing the multimedia bitstreams, and an RDF repository, containing all the necessary metadata (i.e., RDF triples compliant with our model for multimedia bitstreams). The processing layer contains an Adaptation and Packaging Engine (APE) (of which the working has been discussed in Section 4.2), a Structural Metadata Generator (SMG) that enables the creation of triples compliant with the structural part of the model, and an Adaptation Decision-Taking Engine (ADTE). The ADTE calculates adaptation parameters, initializes the APE, and manages the different client sessions [32]. The front-end layer provides access points for clients of the NinSuna platform. New multimedia bitstreams can be uploaded with their corresponding metadata using the authoring service. Information regarding multimedia bitstreams can be obtained using the retrieval service. The adapted and packaged multimedia content is retrieved through the download or streaming service. The streaming service implements the RTSP protocol [3], while the download service makes multimedia content available through (progressive) download-and-play scenarios (e.g., using MP4 as delivery format).

5.1.1 Workflow

Explanatory notes for the workflow within the NinSuna platform (see Figure 7) are given below. The NinSuna platform allows authoring clients to communicate with the authoring service in order to extend the multimedia database. The workflow for uploading new multimedia content to the NinSuna platform is as follows.

(A1) The authoring client sends semantic annotations for a particular multimedia bitstream to the authoring service. These annotations, stored in the RDF repository, consist of RDF triples compliant with the semantic part of the model for multimedia bitstreams.

(A2) The authoring client uploads the actual multimedia content, encoded with a specific coding format, to the multimedia content repository.

(A3) The SMG is used to generate the structural part of the metadata belonging to the uploaded multimedia bitstream. It takes as input the encoded multimedia bitstream and its semantic annotations and produces RDF triples compliant with the structural part of the multimedia model. The RDF triples are subsequently stored in the RDF repository. Note that the semantic metadata are needed by the SMG to create a mapping between the structural and semantic metadata (i.e., to connect random access units with temporal segments using the *hasBitstreamData* property, as discussed in Section 3.1).

Consumer clients of the NinSuna platform can make use of three services: retrieval, download, and streaming. The workflow for retrieving multimedia content is given below.

(R1) The retrieval service makes it possible to query the RDF repository (by making use of SPARQL) in order to browse through the multimedia content, based on the available semantic metadata. Hence, the retrieval service can be compared to a SPARQL endpoint [12] with a number of additional shortcuts to retrieve multimedia bitstreams. Note that only the semantic metadata part can be browsed: the structural metadata are masked for the retrieval service since these metadata are irrelevant for the consumer client.

(R2) The consumer client sends a SPARQL query in order to request a particular multimedia bitstream (for an example of such a query, see Listing 1), together with a description of its usage environment, to the retrieval service which passes this information to the ADTE. The latter creates a new session and selects coding and delivery formats based on the given usage environment characteristics. When the client desires both audio and video, two appropriate coding formats are selected. Next, the ADTE calculates the adaptation parameters for the selected coding formats by matching the scalability information of the requested content with the information about the usage environment (e.g., by comparing the video resolution with the screen size of the end-user device). Note that in the current implementation of our platform, the exchange and interpretation of usage environment information occurs in a trivial way. More specifically, the client indicates the desired coding and delivery format, together with its preferred scalability options (e.g., a frame rate of 15 fps) by means of a proprietary format. Future work may consist of extending the platform with support for a standardized usage environment description format such as the Usage Environment Description (UED) tools that are standardized within MPEG-21 Digital Item Adaptation [23].

(R3) The ADTE initializes one or more APEs (see further) with the received SPARQL query and the calculated adaptation parameters. Furthermore, it creates an URL for the consumer client which indicates where the requested multimedia content can be found. This URL, which contains a session ID, is send back to the consumer client.

The number of APEs that is initialized depends on the number of requested bitstreams and the delivery service used: if only one bitstream is requested, then one APE is initialized; if two or more bitstreams are requested (e.g., a video stream with a corresponding audio stream), then the number of APEs that need to be initialized depends on the requested delivery service (i.e., streaming or download). The latter is illustrated in Figure 8 for a video and corresponding audio stream. If the bitstreams need to be delivered using the streaming service, two APEs are initialized (Figure 8(a)). This is because the streaming service expects one stream of RTP packets for each bitstream. Hence, two separate streams of RTP packets are created for the streaming service (i.e., each stream corresponds to a different RTP session). When the bitstreams are delivered through the download service, only one output stream is expected. Therefore, the selected streams are merged within the APE. More specifically, the two streams are associated with different adaptation processes (i.e., data block selection and

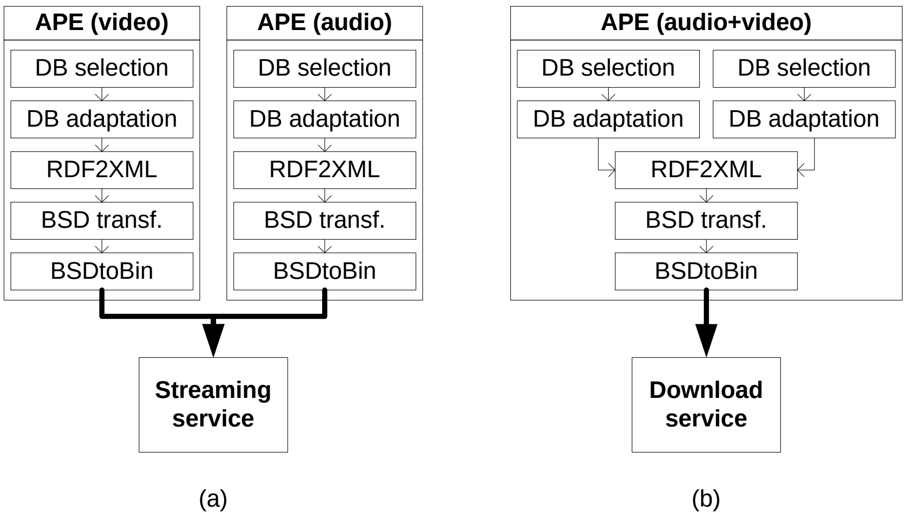


Figure 8: Structural variations of APEs during the adaptation and delivery of corresponding audio and video streams.

adaptation), but these are merged at the beginning of the packaging process (i.e., the RDF-to-XML transformation) (Figure 8(b)). This way, the bitstreams can be packed together in a delivery format targeted for (progressive) download-and-play scenarios (e.g., MP4).

(R4) Dependent on the received URL, the consumer client contacts the progressive download or streaming service to retrieve the desired content. When the streaming service was selected, the consumer client starts an RTP/RTSP session with the NinSuna platform.

(R5) By using the session ID (included in the URL), the download or streaming service contacts the ADTE in order to obtain the proper session. Hence, based on the answer of the ADTE, the download or streaming service can determine which APE(s) will provide the requested content.

(R6) The APEs start working (as described in Section 4.2) and provide the adapted and packaged multimedia bitstream to the download or streaming service. When the streaming service is selected, the APE provides a stream of RTP packets. The latter are sent out by the streaming service to the consumer client according to the RTSP protocol.

(R7) When the usage environment conditions change (e.g., the network connection changes from broadband to smallband), adaptation parameters can be dynamically adjusted. The consumer client uses the retrieval service to announce its new usage environment. The ADTE recalculates and changes the adaptation parameters. As discussed in Section 3.2, to avoid the initialization and evaluation of a new query each time an adaptation property changes, support for dynamic adaptations is provided in the data block transformation step.

5.1.2 Distributing NinSuna across the Network

Communication within the NinSuna platform is realized using the HTTP protocol [4]. Hence, the different components can be distributed across a network. First of all, the different layers (storage, processing, and front-end) can be divided across different machines. Furthermore, a pool of APEs and SMGs can be created to serve a large number of clients. These pools can also be divided across multiple machines. The management of the APEs can be done by the ADTE, which will choose the proper APE based on its current load. Management of the SMGs can be done by the authoring service. Note that such a distributed architecture increases the scalability of the platform, since it allows extending the platform with additional components in order to anticipate an increasing load.

5.1.3 Extensibility

One of the main features of the NinSuna platform is its format-independency. Both the adaptation and packaging process are format-independent. Hence, it is rather straightforward to extend the platform with support for new coding, metadata, and delivery formats. Adding support for a new coding format consists of the following steps.

- A mapping needs to be created between the coding format and the model for multimedia bitstreams. Note that this mapping is actually implemented and performed during the metadata generation step.
- XML transformation filters (implemented in STX or XSLT) need to be created for use during the BSD transformation step. More specifically, delivery formats that support the encapsulation of the new coding format and that are already available in the platform (i.e., a BS Schema exists for the delivery format) need to be taken into account. Hence, for each delivery format that needs to be supported for the new coding format, an XML transformation filter needs to be created.

New metadata formats are inherently supported thanks to the use of our OWL-based model for multimedia bitstreams: it is only necessary to align the new metadata format (i.e., ontology) with the semantic part of the model. Finally, a new delivery format is supported by taking the following steps.

- A BS Schema needs to be created, which describes the high-level syntax structures and syntax elements of the delivery format (as discussed in Section 4.2).
- XML transformation filters (implemented in STX or XSLT) need to be created for use during the BSD transformation step. More specifically, coding formats that are already supported by the platform and that are allowed to be incapsulated in the new delivery format need to be taken into account. Hence, for each coding format that needs to be supported for the new delivery format, an XML transformation filter needs to be created.

5.2 Implementation

The Java Platform, Enterprise Edition (Java EE) is used to implement NinSuna. Java EE enables the development of robust and scalable server-side Java applications; furthermore, complete web services support is available. Sesame (version 2.1), which is an open source RDF database with support for RDF Schema inferencing and querying, is used as RDF repository. The Sesame RDF API is used to access the repository and to evaluate the SPARQL queries. Within the APE, Saxon 6.5.5 and Joost v.2008-05-28 are used as XSLT and STX transformation engine respectively. Also, an own Java implementation of the BSDtoBin parser is used. Note that we did not use the BSDL reference software due to a lack of support for multithreading. The SMG consists of parsers generated by Flavor [17], enhanced with support for the generation of RDF triples compliant with the structural part of the model for multimedia bitstreams. Finally, the streaming service uses the RTSP implementation available in the C++ library of Live555 Streaming Media.

5.3 Performance Measurements

In this section, a number of performance measurements are presented to provide the reader with an impression of the performance of the NinSuna platform. First, a use case scenario is discussed, which is then followed by the experimental results.

5.3.1 Use Case Scenario

A number of news sequences were used to test our adaptation and delivery platform. Semi-automatic annotation was used for each news sequence, i.e., shots were automatically detected after which each detected shot was manually annotated by a number of keywords. When mapping these metadata to our model for multimedia bitstreams, each shot corresponds to a *TemporalSegment* which contains a *keyword* property (values for this property correspond to the keywords).

		Video		Audio	
Name	Length (s)	Size (MB)	Bit rate (MBit/s)	Size (MB)	Bit rate (KBit/s)
news1	1302	217.5	1.34	19.7	124
news2	1301	198.7	1.22	19.4	122
news3	1274	184.7	1.16	19.9	128
news4	1284	196.9	1.23	20.0	128
news5	1460	198.2	1.09	22.3	125
news6	1269	187.3	1.18	19.2	124
news7	1305	174.4	1.07	20.4	128

Table 1: Overview of the bitstream characteristics.

The scenario to obtain (parts of) a news sequence is as follows. The user searches, based on keywords, for news sequences containing news topics that are of his/her particular interest. Next, the user requests the selected news scenes and provides a description of the usage environment to the NinSuna platform. The latter selects the requested audio and video scenes, performs structural adaptations if needed (i.e., exploitation of scalability such as frame rate scaling), and packages the selected streams.

The multimedia content archive of NinSuna contained seven news sequences, each having a resolution of 720x432, a frame rate of 25 fps, and a length of approximately 22 minutes. The video streams of the news sequences were encoded using H.264/AVC. A hierarchical coding structure was used to obtain three layers of temporal scalability (i.e., the videos can be rescaled from 25 fps to 12.5 fps and 6.25 fps) [13]. Instantaneous Decoding Refresh (IDR) frames were inserted every 16 frames to obtain feasible random access. Further, the audio streams of the news sequences were encoded using AAC (with a sampling frequency equal to 48000). Additional bitstream characteristics can be found in Table 1.

Two delivery formats are available in our scenario: RTP (streaming service) and MP4 (download service). Hence, three XML transformation stylesheets were developed: one STX stylesheet to guide the packaging of an H.264/AVC stream into RTP packets, one STX stylesheet to guide the packaging of an AAC stream into RTP packets, and one XSLT stylesheet to guide the packaging of an H.264/AVC and AAC stream into an MP4 container. We use STX for RTP because of its streaming capabilities. Note that our RTP STX stylesheet implements the different RTP packetization modes for H.264/AVC (i.e., Fragmentation Unit (FU), Multi-Time Aggregation Packet (MTAP), and Single-Time Aggregation Packet (STAP)) [5]. The latter is only possible if the data blocks are accompanied by a delivery parameter representing the Network Abstraction Layer Unit (NALU) type. Hence, thanks to this delivery parameter, we can determine the beginning of a NALU and its type, which is necessary to implement the packetization modes of the RTP format for H.264/AVC. Furthermore, the Sequence and Picture Parameter Sets are transmitted through the Session Description Protocol (SDP, [6]). Within the streaming service, the SDP descriptions are sent within the reply of the client's RTSP DESCRIBE message. Note that such SDP descriptions are also created within the RTP packaging filter for H.264/AVC.

XSLT is used for MP4 because the format defines header values containing information related to the whole multimedia bitstream, and these values need to be calculated at runtime (e.g., length of the resulting MP4 file and random access points in the multimedia bitstreams). Note that we could have also used STX to implement the packaging process for MP4, but that would have introduced a significant overhead in terms of implementation effort because of the streaming character of STX.

5.3.2 Experimental Results

Performance measurements were done on a PC having an Intel Pentium D 2.8 GHz CPU and 1 GB of system memory at its disposal. The operating system used was Windows XP Pro SP2, running Java 2 Runtime Environment (SE version 1.5.0_09). JProfiler 5.1.4 was used to profile our platform components. All time measurements were executed six times, whereupon an average was calculated over the last five runs to avoid startup effects.

	Video			Audio		
Name	Time (s)	Speed (MBit/s)	# Data blocks	Time (s)	Speed (KBit/s)	# Data blocks
news1	695.8	2.5	32561	398.3	405.5	61051
news2	753.0	2.1	32538	381.8	417.0	61008
news3	634.2	2.3	31875	395.7	412.0	59765
news4	625.1	2.5	32111	396.2	413.4	60207
news5	779.8	2.0	41008	474.4	385.3	76889
news6	736.6	2.0	31729	403.0	391.3	59491
news7	659.7	2.1	32638	392.5	425.6	61196

Table 2: Execution times for the generation of structural metadata.

Structural Metadata Generation

As discussed in Section 5.1, the SMG enables the creation of triples compliant with the structural part of the model for multimedia bitstreams. Enhanced Flavor-based [17] parsers are used to implement the SMG. For each of the seven news sequences, the SMG is used to generate their structural metadata. The memory usage of the SMG is low and constant (approximatly 3 MB). Its execution times are provided in Table 2. For all multimedia bitstreams (audio and video streams), the SMG is able to generate the structural metadata in real time (i.e., the execution speed is higher than the bit rate of the multimedia bitstream). The execution speed of the SMG is dependent on the following parameters.

- *# parse units per second*: a higher number of parse units per second implies a lower execution speed for the SMG. Note that the number of parse units per second is dependent on the coding format (and its encoding parameters). For instance, a parse unit in H.264/AVC corresponds to a NALU. We have encoded the seven video news sequences in such a way that each NALU corresponds to one frame. Hence, the video streams are characterized by 25 parse units per second. Further, parse units for the audio news sequences correspond to AAC frames (containing 1024 samples), implying that the audio news sequences are characterized by 46.9 parse units per second.

- *# skipped bytes per parse unit*: a higher number of skipped bytes per parse unit implies a higher execution speed for the SMG. These skipped bytes correspond to the coded (audio or video) data (e.g., motion vectors and transform coefficients), because the SMG only parses high-level syntax structures of a multimedia bitstream. In our example, the video news sequences have a higher bit rate than the audio news sequences (see Table 1), implying that the video streams contain more coded data and hence that more bytes can be skipped by the SMG.

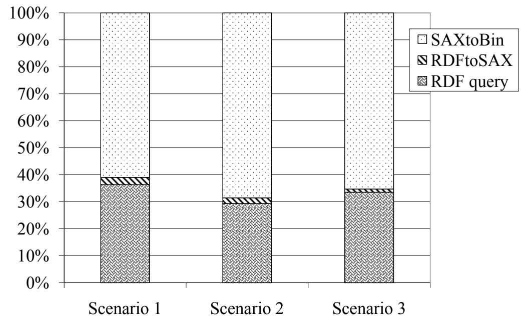
Table 2 also shows the number of data blocks that is generated for each multimedia bitstream. In our case, a data block corresponds to a parse unit (i.e., a NALU for H.264/AVC and a frame for AAC). Since each data block is represented as an RDF graph, we can calculate the number of RDF triples that is necessary to represent the structural metadata. One RDF data block graph consists of 5 RDF triples. Further, one RAU (consisting of 3 RDF triples) points to 16 data blocks (for video) or 30 data blocks (for audio). For example, the total amount of RDF triples to represent the structural metadata for the *news1* video sequence is equal to $(32561*5)+((32561/16)*3) = 168910$.

Delivery of News Fragments

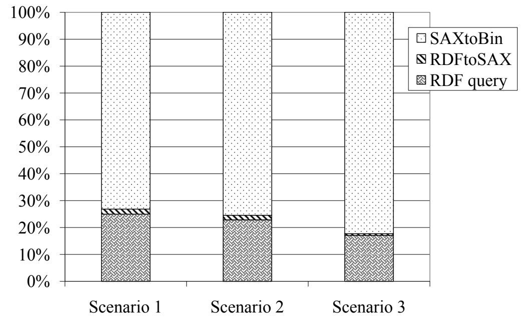
Three scenarios are considered to evaluate the delivery of (partial) multimedia bitstreams. In the first scenario, a fragment of the multimedia bitstream is selected occuring in the beginning of the multimedia bitstream. The second scenario takes a fragment at the end of the multimedia bitstream. The third scenario takes both fragments of scenarios 1 and 2. We distinguish these three scenarios in order to be able to investigate the impact of the temporal location of fragments

	Original start offset (s)	Fragment length (s)	Fragment size (MB)		Peak memory usage (MB)		Latency (s)	
			MP4	RTP	MP4	RTP	MP4	RTP
Scenario 1	18	128	11.8	12.0	24	10	2.4	0.2
Scenario 2	810	139	18.5	18.0	25	10	2.5	0.2
Scenario 3	18	267	30.3	30.8	37	10	8.2	0.2

Table 3: Characteristics of the three delivery scenarios applied to the news2 sequence.



(a) News fragments delivered using RTP.



(b) News fragments delivered using MP4

Figure 9: Proportion in terms of execution time percentages between components of the NinSuna platform.

in the original bitstream and the length of the selected fragments. More information regarding the resulting multimedia fragments of the *news2* audio and video sequences is provided in Table 3.

We have evaluated the media adaptation and delivery process within NinSuna in terms of peak memory consumption and execution times. Regarding the peak memory consumption, we do not consider the memory usage of the Java Virtual Machine and the Java Application Server, i.e., only the memory usage is measured for the APE (i.e., data block selection and adaptation, RDF-to-XML transformation, and BSD transformation and BSDtoBin). As shown in Table 3, RTP delivery is characterized by a low and constant memory usage (i.e., 10 MB). On the contrary, delivery using MP4 introduces memory usage that is dependent on the length (in terms of data blocks) of the multimedia fragments. This is due to the XSLT transformation that needs to store the full XML document, resulting from the RDF-to-SAX transformation, in memory. Note that this is necessary due to the presence of headers occurring in front of the MP4 file and covering information regarding the full bitstream (e.g., a list of random access points). In future developments, the XSLT stylesheet should be replaced by a more efficient streaming XML filter. Also in Table 3, the time between the client's request and the first delivered byte (measured at server-side to avoid network delay) is provided (i.e., the latency). For RTP, the latency is low and independent of the length and position of the media fragment (i.e., 0.2 s). For MP4, the latency is dependent on the length of the requested media fragment. For instance, for scenario 3, the resulting MP4 file is available for (progressive) download after 8.2 s.

In Figure 9, the proportion between components of the APE are shown in terms of execution time percentages. RDF query, RDFtoSAX, and SAXtoBin correspond to data block selection, data block adaptation and RDF-to-XML transformation, and BSD transformation and BSDtoBin respectively. For RTP delivery, SAXtoBin takes most of the time with 65 % on average. RDFtoSAX only requires a small proportion in terms of execution time (2 %), while RDF query takes 33 %. These proportions are independent of the length of the resulting media fragment, i.e., they are only influenced by the resulting bit rate of the media fragment. For MP4 delivery, the proportions between the APE components are not independent of the length of the media fragment (the longer the fragment, the more time is spent by SAXtoBin). This is due to the decreasing performance of the BSD transformation when the size of the incoming XML document, which is dependent on the length of the media fragment, increases.

6 Directions for Future Research

Future work consists of the development of a more efficient MP4 packetizing filter. A SAX filter could be created which is able to keep the header data of the MP4 file in memory, while already writing the payload data to a temporary file. Note that the MP4 format also provides a solution in the form of *fragments*. More specifically, instead of one header covering the whole MP4 file, fragmented MP4 files contain multiple headers only covering parts of the MP4 file. This way, MP4 files can be created in an 'on-the-fly' fashion, making it no longer necessary to collect information regarding the full bitstream to create the header. Unfortunately, fragmented MP4 files are not supported yet by most media players.

Further, optimizations for the storage and retrieval of the structural metadata should be investigated. For example, one solution for this problem is to store the structural metadata and scalability information in an RDF store which is specifically designed for the model for multimedia bitstreams. In particular, the structural metadata and scalability information can be stored in a highly scalable Relational Database Management System (RDBMS), using a database scheme based on the structural and scalability part of the model for media bitstreams. Hence, such a RDBMS can be seen as an efficient RDF store specifically designed for our model. RDBMSs should be capable of dealing with a large amount of structural metadata since they are mature, stable, and scalable, while also providing a high performance in terms of query execution speed.

Other future work consists of linking our model for multimedia bitstreams to other multimedia ontologies. This will allow us to add fully detailed annotations to multimedia bitstreams. Existing multimedia ontologies are listed by the MultiMedia SEMantics (MMSEM) W3C Incubator Group [7]. Following MMSEM, W3C has started a Media Annotation Working Group[6] which has as mission to provide an ontology designed to facilitate cross-community data integration of information related to media objects in the web, such as video, audio, and images.

Finally, additional topics for future research are the addition of support for the UED tools of MPEG-21 DIA, extending the model for multimedia bitstreams to support spatial fragment selection, and the development of a fully semantic-aware multimedia browser, which will serve as a consumer client for the NinSuna platform.

7 Discussion and Conclusions

In this paper, we presented a format-independent multimedia content adaptation and delivery platform, based on a model for multimedia bitstreams. This model covers the structural, semantic, and scalability properties of multimedia bitstreams and is implemented by making use of OWL. Further, it provides support for a seamless integration of adaptation operations and semantic metadata, and supports format-independent packaging of multimedia content. Multimedia adaptation is performed by selecting and adapting portions of the structural metadata using SPARQL. Multimedia packaging is obtained by encapsulating the selected and adapted structural metadata within a specific delivery format. This packaging process is implemented using XML transformation filters and MPEG-B BSDL.

In previous work by the authors [42], a fully integrated multimedia adaptation platform relying on XML-driven content adaptation engines was presented. However, as discussed in Section 2, semantic adaptations introduce difficulties due to interoperability problems of XML between different metadata standards. Furthermore, the multimedia delivery in [42] is implemented by means of a dedicated and coding-format specific streaming server. Hence, no generic solution for multimedia packaging is present. The NinSuna platform provides solutions for these problems by offering a seamless integration between adaptation processes and metadata standards, and by providing fully format-independent multimedia content adaptation and packaging engines.

The Continuous Media Markup Language (CMML, [35]) allows to annotate and index continuous media files. The presented architecture is able to extract temporal segments of media resources using a temporal URI scheme. A new file format is presented (i.e., Annodex), which enables encapsulation of any type of streamable media resource (i.e., Annodex is coding-format independent).

[6]See `http://www.w3.org/2008/01/media-annotations-wg.html` for more information.

Annodex is based on the Ogg encapsulation format and is basically a bitstream consisting of multimedia bitstreams combined with a CMML file. Comparing Annodex to our approach presented in this paper, we can state that both solutions allow to extract temporal fragments from media resources in a format-independent manner. However, Annodex requires the use of a single delivery format (i.e., Ogg) while our approach is able to deliver media content using any delivery format, in a format-independent way. Further, in contrast to our approach, Annodex does not support structural adaptations of media resources (i.e., exploitation of scalability layers).

Digital Item Streaming (DIS, [24]) is part 18 of MPEG-21 and enables the incremental delivery of a Digital Item (covering both metadata and media resources) in a piece-wise fashion. DIS relies on the Bitstream Binding Language (BBL, [41]) for this purpose. BBL defines syntax and semantics to describe instructions on how a Digital Item can be fragmented and mapped into one or more delivery channels. It uses the same principles for serializing the packed multimedia bitstream as our proposed method, i.e., MPEG-B BSDL is used to abstract the multimedia bitstream and to enable the use of format-agnostic software modules. However, the BBL approach requires a new language to be used to specify the fragmentation and packetization process. Our proposed method to perform format-independent packaging only requires knowledge of commonly used XML transformation languages such as XSLT or STX. Furthermore, our model for multimedia bitstreams provides support for the multimedia packaging process (i.e., timestamp support and coding-format specific parameters). Hence, this information can already be calculated during the metadata generation step, which is in contrast to the BBL approach where this information needs to be calculated during the packaging process.

Ransburg *et al.* propose to use *Media Streaming Instructions* within BSDs to implement a generic streaming server [37]. More specifically, access units (i.e., the smallest unit of data to which timing may be attached) are identified and timestamps are assigned to them. Note that these Media Streaming Instructions have been adopted in the second amendment of the MPEG-21 DIA specification [25]. Using Media Streaming Instructions, the fragmentation process and timestamp calculation is performed during the BSD generation step (i.e., during structural metadata generation). However, the fragmentation process is dependent on the delivery format (e.g., fragmentation of H.264/AVC streams is different for RTP and MP4 packetization). Also, BSDs including Media Streaming Instructions are processed by delivery-format specific software modules (e.g., an RTP packetizer).

Finally, the Darwin Streaming Server[7] (DSS) is an open source, cross-platform RTP/RTSP streaming server. It provides a coding-format agnostic design, i.e., no codecs are present in the server. The streaming of media resources is guided by hint tracks, which contain all the information necessary to packetize and stream the media resource. Note that the creation of these hint tracks is coding-format specific (e.g., MP4Box[8] is a commonly used tool for the creation of hint tracks). Hint tracks can be compared to a part of our structural metadata (i.e., the mapping of timestamps to byte ranges of the media resource). However, support for adaptation operations is not available in DSS. Also, packing multimedia content with other delivery formats (other than RTP) is not possible.

Acknowledgments

The research activities as 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), and the European Union.

References

[1] RFC 3550, "RTP: A Transport Protocol for Real-Time Applications," Available on `http://www.ietf.org/rfc/rfc3550.txt`.

[7]`http://developer.apple.com/opensource/server/streaming/`

[8]`http://gpac.sourceforge.net/packager.php`

[2] "SUMO: Suggested Upper Merged Ontology," Available on http://www.ontologyportal.org/.

[3] RFC 2326, "Real Time Streaming Protocol," Available on http://www.ietf.org/rfc/rfc2326.txt.

[4] RFC 2616, "Hypertext Transfer Protocol – HTTP/1.1," Available on http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[5] RFC 3984: "RTP Payload Format for H.264 Video," Available on http://www.ietf.org/rfc/rfc3984.txt.

[6] RFC 2327: "SDP: Session Description Protocol," Available on http://www.ietf.org/rfc/rfc2327.txt.

[7] W3C Multimedia Semantics Incubator Group. Available on http://www.w3.org/2005/Incubator/mmsem/.

[8] M. Amielh and S. Devillers. Multimedia Content Adaptation with XML. In *Proceedings of 8th International Conference on Multimedia Modeling*, pages 127–145, Amsterdam, The Netherlands, November 2001.

[9] M. Amielh and S. Devillers. Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content Adaptation. In *Proceedings of 11th International World Wide Web Conference*, Honolulu, Hawaii, May 2002. Available on http://wwwconf.ecs.soton.ac.uk/archive/00000185/01/index.html.

[10] R. Arndt, R. Troncy, S. Staab, L. Hardman, and M. Vacura. COMM: Designing a Well-Founded Multimedia Ontology for the Web. In *6th International Semantic Web Conference (ISWC 2007)*, Busan, Korea, November 2007.

[11] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG2000 Still Image Coding System: an Overview. *IEEE Trans. Consumer Electron.*, 46(4):1103–1127, November 2000.

[12] K. Clark, L. Feigenbaum, and E. Torres, editors. *SPARQL Protocol for RDF.* W3C Recommendation. World Wide Web Consortium, January 2008.

[13] W. De Neve, D. Van Deursen, D. De Schrijver, K. De Wolf, and R. Van de Walle. Using Bitstream Structure Descriptions for the Exploitation of Multi-layered Temporal Scalability in H.264/AVC's Base Specification. *Lecture Notes in Computer Science*, 3768:641–652, November 2005.

[14] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.

[15] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner. Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments. *IEEE Trans. Multimedia*, 7(3):463–470, June 2005.

[16] Dublin Core Metadata Initiative. Dublin Core Metadata Element Set, version 1.1: Reference Description. Technical report, Dublin Core Metadata Initiative, 2004. Available on http://www.dublincore.org/documents/dces/.

[17] A. Eleftheriadis. Flavor: A Language for Media Representation. In *Proceedings of ACM Multimedia Conference*, pages 1–9, Seattle, WA, November 1997.

[18] M. M. Hannuksela, Y.-K. Wang, and M. Gabbouj. Isolated Regions in Video Coding. *IEEE Trans. Multimedia*, 6:259–267, April 2004.

[19] D. Hong and A. Eleftheriadis. XFlavor: providing XML features in media representation. *Multimedia Tools and Applications*, 39(1):101–116, 2008.

[20] ISO/IEC. 23000-5: Information technology - MPEG-B Systems technologies Part 5: Bitstream Syntax Description Language (BSDL).

[21] ISO/IEC. 15938-5:2003 Information technology – Multimedia content description interface – Part 5: Multimedia description schemes, September 2003.

[22] ISO/IEC. 14496-14:2003 Information technology – Coding of Audio, Picture, Multimedia and Hypermedia Information – Part 14: MP4 file format, February 2004.

[23] ISO/IEC. 21000-7:2004 Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation, October 2004.

[24] ISO/IEC. 21000-18:2007 Information technology – Multimedia framework (MPEG-21) – Part 18: Digital Item Streaming, June 2007.

[25] ISO/IEC. Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation, Amendment 2: Dynamic and Distributed Adaptation. ISO/IEC 21000-7:2007/FPDAmd 2, January 2007.

[26] ISO/IEC JTC 1. Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio. ISO/IEC 11172-3:1993, 1993.

[27] ISO/IEC JTC 1. Information technology – Generic coding of moving pictures and associated audio information: Video. ISO/IEC 13818-2:2000, 2000.

[28] ISO/IEC JTC 1. Information technology – Coding of audio-visual objects – Part 3: Audio. ISO/IEC 14496-3:2005, 2005.

[29] ITU-T and ISO/IEC JTC 1. Advanced Video Coding for Generic Audiovisual Services. ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, 2003.

[30] G. Klyne and J. J. Carroll, editors. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. World Wide Web Consortium, February 2004.

[31] D. McGuinness and F. van Harmelen, editors. *OWL Web Ontology Language: Overview*. W3C Recommendation. World Wide Web Consortium, February 2004.

[32] D. Mukherjee, E. Delfosse, J.-G. Kim, and Y. Wang. Optimal Adaptation Decision-taking for Terminal and Network Quality-of-service. *IEEE Trans. Multimedia*, 7(3):454–462, June 2005.

[33] J.-R. Ohm. Advances in scalable video coding. *Proc. IEEE*, 93(1):42–56, January 2005.

[34] G. Panis, A. Hutter, J. Heuer, H. Hellwagner, H. Kosch, C. Timmerer, S. Devillers, and M. Amielh. Bitstream Syntax Description: a Tool for Multimedia Resource Adaptation within MPEG-21. *Signal Processing: Image Communication*, 18(8):721–747, September 2003.

[35] S. Pfeiffer, C. Parker, and C. Schremmer. Annodex: A Simple Architecture to Enable Hyperlinking, Search & Retrieval of Time Continuous Data on the Web. In *MIR '03: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, pages 87–93, Berkeley, California, November 2003.

[36] E. Prud'hommeaux and A. Seaborne, editors. *SPARQL Query Language for RDF*. W3C Recommendation. World Wide Web Consortium, November 2007.

[37] M. Ransburg, S. Devillers, C. Timmerer, and H. Hellwagner. Processing and Delivery of Multimedia Metadata for Multimedia Content Streaming. In *Proceedings of 6th Workshop on Multimedia Semantics - The Role of Metadata*, Aachen, Germany, March 2007.

[38] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Trans. Circuits Syst. Video Technol.*, 17(9):1103–1120, September 2007.

[39] SMPTE. Material Exchange Format (MXF) – File Format Specification (Standard). SMPTE 377M-2004, 2004.

[40] S. Srinivasan, C. Tu, S. L. Regunathan, and G. J. Sullivan. HD Photo: a New Image Coding Technology for Digital Photography. In *Proceedings of the SPIE*, volume 6696, San Diego, US-CA, USA, August 2007.

[41] J. Thomas-Kerr, I. Burnett, and C. Ritz. Format-Independent Rich Media Delivery Using the Bitstream Binding Language. *IEEE Trans. Multimedia*, 10(3):514–522, April 2008.

[42] D. Van Deursen, S. De Bruyne, W. Van Lancker, W. De Neve, H. Hellwagner, and R. Van de Walle. MuMiVA: a Multimedia Delivery Platform using Format-agnostic, XML-driven Content Adaptation. In *Proceedings of the 9th International Symposium on Multimedia*, pages 131–138, Taichung, Taiwan, december 2007.

[43] A. Vetro, C. Christopoulos, and T. Ebrahimi. Universal Multimedia Access. *IEEE Signal Processing Mag.*, 20(2):16, March 2003.