ABSTRACT
The delivery and the adaptation of multimedia content in distributed and heterogeneous environments require flexible control and management mechanisms in terminals and/or in control entities inside the network to enable terminal and application mobility. It is important to reach interoperability between the IETF approaches and the MPEG-21 DIA efforts for multimedia streaming and adaptation to cope with the aforementioned requirements. MPEG-21 Digital Item Adaptation (DIA) provides normative descriptions for supporting the adaptation of multimedia content, but it does not define interactions with existing transport and control technologies, in order to remain independent of other technologies. On the other hand, the IETF standardization work on session descriptions (SDP and SDPng) provides the necessary transport and control mechanisms. By combining the MPEG-21 DIA and SDPng formats (both designed in XML), we create a converged model that enables the integration of session management and negotiation protocols (e.g., SIP or Megaco/H.248). This work presents a possible implementation of the SDPng and MPEG-21 DIA convergence as well as concepts and implementation of network-based content adaptation through media gateways that enable a flexible multimedia management for heterogeneous consumer terminals.

KEY WORDS
MPEG-21 DIA, SDP, SDPng, QoS, content adaptation, media gateways

1. Introduction
The deployment of advanced multimedia management systems requires advanced description and control models for the media. MPEG-21 Digital Item Adaptation (DIA) [1] provides normative XML descriptions (designated as tools) for applications that handle multimedia contents (named also the Digital Items) by shielding users from network and terminal installation, management and implementation issues. Digital Items are specified as a bundle of resources and (associated) metadata within a standardized representation (i.e., structure) for various purposes that are integrated in a generalized adaptation framework. However, MPEG-21 does not specify relations to existing technologies for transport mechanisms, in order to be independent of other specifications and open for future developments. On the other hand, Session Description Protocol next generation (SDPng) [2] defines a language for describing multimedia sessions with respect to technology-specific configuration parameters, terminals, conference services, media gateways, etc.

This contribution presents a practical approach for combining SDPng with MPEG-21 DIA descriptions. Both MPEG-21 DIA and SDPng description formats adopt the eXtensible Markup Language (XML) [3], thus, a smooth integration of the two technologies is provided. In this paper, we propose a generalized session control mechanism that applies the integrated version of SDPng and MPEG-21 DIA components, and may be reused for backward compatibility with the older Session Description Protocol (SDP) [4]. Furthermore, an adaptation management mechanism for multimedia that is able to fulfil the requirements of heterogeneous clients and which incorporates distributed content delivery and adaptation based on our generalized session control model is presented. We discuss how existing mechanisms should be extended in order to provide an optimised management model for adaptive multimedia delivery.

2. Background
SDP [4] and its successor SDPng [2] are meta-protocols that describe other protocols and configuration mechanisms of services and applications in multimedia sessions. Usually, the respective descriptions are carried inside other session layer protocols (like SIP [5], RTSP [6] or Megaco [7]), to exchange for instance configuration information pertaining to RTP-based multimedia streams [8]. The main reason for the transition from SDP to SDPng is the adoption of the XML format for session descriptions that shall enable extensible representation mechanism and higher flexibility at introducing new defi-
nitions. In addition to the already available mechanisms for codec descriptions and RTP configurations [9], SDPng shall enable negotiations of various Quality-of-Service (QoS) parameters associated with content consumers as well as with content providers.

MPEG-21 provides a multimedia framework enabling augmented and transparent use of multimedia resources across a wide range of terminals and networks [10]. Therefore, the concept of so-called Digital Items (DIs) that contain information about their own adaptability is introduced. This enables DIs to be adapted on the fly on end-terminals (e.g., on a video server) as well as on network nodes if given conditions are fulfilled, e.g., QoS aspects, usage environment properties or user preferences.

MPEG-21 DIA specifies the syntax and semantics of normative descriptions that are used to assist the adaptation of DIs. Users can apply these descriptions to satisfy transmission, storage and consumption constraints, as well as QoS management [1]. The adaptation process in DIA follows a generic concept for multimedia adaptation utilizing Bitstream Syntax Descriptions (BSDs) in order to remain codec agnostic [11]. Considering these technologies, the aim of this work is to produce a consolidated model of MPEG-21 DIA and SDPng under the name SDPng++ that ensures backward compatibility with applications using SDP.

The XML specification of SDPng++ is already described in several works [12] – [15]. Hence, we present here only the major features of this specification used for the implementation. SDPng++ reuses the basic SDPng XML namespace [13] and extends SDPng’s RTP-Package for specifying streaming media that can be tracked for QoS-guarantee purposes on the network [15]. SDPng++ integrates MPEG-21 DIA Usage Environment (UED) and Bitstream Syntax Description (BSD) namespaces into the SDPng XML schema to provide mechanisms for QoS constraints descriptions at network and application level (not available through the basic SDPng) and to accommodate MPEG specific bitstream adaptation for multimedia payloads in RTP streams [13][14].

3. Description and Control Model for the Multimedia

This section presents the description and control model for the multimedia based on SDPng and MPEG-21 DIA.

3.1 Session control model with SDP and SDPng++

SDP uses an ABNF-like format [16] that has a restricted depth of the definition hierarchy. SDPng applies XML and non-limited depth of the definition hierarchy in addition to the distinction between simple variables (i.e., attributes) and object variables (i.e., elements). Hence, this may lead to structural mismatches and lack of interpretation expressiveness in places where the applications’ internal object-model specifies leaves of the description hierarchy, e.g., if SDP defines a mapping from a single key to a simple value mapping, SDPng will require for the same definition a key-to-object mapping for objects with a specific structure, multiple contained values and multiple identifiers for these values.

In order to cope with this definition problem, we propose a format-independent model for session descriptions as depicted in Figure 1. The model represents the natural hierarchy of an adaptive multimedia session, i.e., the multimedia session contains one or multiple streams and each stream can be represented by one or multiple codecs.

Every session, stream and codec object contains an identifier (e.g., session type, number of the stream and codec name) and can be associated with one or multiple Info, Constraint and/or Parameter objects. These objects are represented by an identifier and a general object (e.g., java.lang.Object [17]). The general object value makes thus the representation applicable for alternative description models. It is up to the application to determine and interpret the object value appropriately, for instance upon the MIME-type of the description format specified in the description carrier (see [5] – [7]).

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter</th>
<th>Constraint</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP</td>
<td>Session</td>
<td>“w=” line</td>
<td>“a=” line</td>
</tr>
<tr>
<td>Stream</td>
<td>“m=” line</td>
<td>“c=” line</td>
<td>“b=” line</td>
</tr>
<tr>
<td>Codec</td>
<td>“a=” line</td>
<td>“a=rtpmap” line</td>
<td>“b=” line</td>
</tr>
</tbody>
</table>

SDPng++

| Session | application/ sdpg MIME-type | <sdpg> header element | <m21-dia: AvailableBandwidth> element | <sdpng-dia: BSDLiveLink> element |
| Stream | <component> element | <rtp:udp> element | <m21-dia: AvailableBandwidth> element | <sdpng-dia: BSDLiveLink> element |

Table 1:

SDP and SDPng++ mapping to general session description model – an example
Figure 1: General definition and control model

Table 1 shows an example on how SDP and SDPng++ elements can be mapped to objects as defined in the general definition and control model. Due to the flat format of the SDP description model [4], there is no explicit separation of the input data into parameters, constraints and info objects. However, the semantics of the SDP lines enable such specific interpretation. Within the SDPng [2] (respectively SDPng++ [13][14]) model, there is a specific association between the XML elements and the three logical objects of the general model. The differentiation is expressed through the XML top-level elements as defined in SDPng (i.e., `<cap>`, `<def>`, `<cfg>`, `<constraints>` and `<info>`). In the general model the information contained in the `<cap>`, `<def>` and `<cfg>` elements is interpreted as Parameters. The remaining two SDPng elements correspond one-to-one with the objects in the logical description model. The ID information for the SDP and SDPng sessions, streams and codecs corresponds either to an implicit entry or to explicit naming of the respective session, stream and codec. For instance the position of any “m=” line in SDP document interpreted as number is an implicit ID and the SDPng name attribute within an element an explicit one.

3.2 SDPng++ implementation

The current implementation of the SDPng++ comprises the SDPng core XML management using Java 1.4.x. The SDPng XML processing applies SAX parsing with the SAX interface for Java [18] and Apache SAX parser Xerces-J [19]. The parsing of SDPng++ requires SAX in order to speed up the application processing as the SDPng++ code is used together with session control protocols that imply stringent time limitations. Furthermore, the application of SAX reduces the number of empty objects in the XML-object tree compared to the DOM processing [20]. The DOM generates objects for all the elements specified through the XML Schema and produces empty objects for all elements that are not used in the instance document (e.g., in the actual SDPng++ XML description). Currently the SDPng++ displays non-optimised XML conversion between SDPng and MPEG-21 DIA. Hence, the resulting DOM-based XML object tree will be inefficient, since the schema contains some descriptions that are currently not in use for protocol purposes. For instance, some of the MPEG-21 UED definitions characterize local application environments but are useless in distributed environments thus these definitions do not need to be negotiated via the utilized session protocol (see also [13][14]).

Figure 2: SDPng++ XML processing

Figure 3: SDPng++ GUI Screen

Table 2: Parser and Generator scaling behaviour

<table>
<thead>
<tr>
<th>Input size [XML lines]</th>
<th>Average Processing Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parser</td>
</tr>
<tr>
<td>100</td>
<td>17.33</td>
</tr>
<tr>
<td>500</td>
<td>51.42</td>
</tr>
<tr>
<td>1000</td>
<td>105.76</td>
</tr>
</tbody>
</table>
The major elements of the SDPng core XML management are (Figure 2):

- The **Parser** implements SAX API for Java and reuses XERCES-J. It generates non-specific data objects named `DescriptionComposite` that can represent any XML element with a value and attributes. The class `DescriptionComposite` inherits the abstract class `Description` that serves as an accumulator for the information within a single XML element (i.e., name, value as well as its attribute names and values). The `Description` provides abstract methods for the management of the information contained in an XML element. These methods are fully implemented within the `DescriptionComposite`. The Parser builds an object tree out of the generated Composite objects that corresponds to the XML tree in an SDPng++ message. The linking of the XML objects is provided through the `DescriptionComposite` mechanisms for accessing its direct parent and children objects (i.e., a `DescriptionComposite` object has only one parent that is also a Composite and one or many children Composites). For the linking purposes, a `DescriptionComposite` is able to accumulate one or multiple `Description` objects.

- The **Generator** is a customized generator for XML code that is designed to produce any XML output document. It uses specific representations for the XML elements implemented in Java that correspond to the XML Schema definitions of SDPng++ (see [13][14]). Every single object type corresponds to every single SDPng XML element of the schema (Figure 2 shows an example of some of the elements applied in the SDPng++ schema). Each of the specific objects is implemented as a single class that extends the `DescriptionComposite` in order to inherit the object-tree building mechanisms provided by the `DescriptionComposite`. Unlike the `DescriptionComposite`, every specific element-mapping object corresponds to just one element specified through the SDPng XML schema and provides specific management per defined attributes of the element and per defined children of the element using **Facade Design Pattern** (see [21]). The specific mapping is expressed the following way, e.g., for accessing the maximum bandwidth information the specific element-mapping object `AvailableBandwidthElement` provides a method `getMaxBandwidth()` whereas the information in the `DescriptionComposite` is accessed through the application of combinations of classes and their methods, like `composite.getAttributes()` that returns a `java.util.Hashable` and the Hashtable has to be questioned then about containing a key named `maximum` (see also the GUI example in Figure 3). The specific element-mapping objects provide structures that are closer in their interpretation to the general definition and control model shown in Section 3.1 and these structures can be directly used by the adaptive application. The specific objects define also object-to-XML serialization mechanism for the XML element that the object represents. The Generator applies the objects serialization mechanism to produce a valid SDPng XML document, but as the Generator does not understand the XML schema, it uses the `CompositeToSchemaObjectMapper` in order to provide validation for the resulting XML document.

The same XML processing model is used in [22] and some preliminary measurements thereof are shown in Table 2. The XML lines used for the measurements represent the text between two XML indentations (i.e., “<” and “>” separator symbols). The measurements were conducted on a PC equipped with a 700 MHz AMD Duron processor and 256 MB of memory. The relatively low processing power of the PC is intentionally chosen to emulate mobile terminal processing behaviour. Nevertheless, the processing procedures display good scalability.

### 4. Advanced Multimedia Content Adaptation

The MPEG-21 DIA provides means for enabling device- and coding-format independence of media contents [23]. We assume that multiple adaptation services sharing metadata will apply standards provided by the different standardisation bodies including MPEG, as it is not realistic to expect that a single adaptation engine is capable to handle all kinds of usage environments and coding formats that will emerge on the market. An example of such an adaptation service is depicted in Figure 4. In this example, User B (the consumer) requests a Digital Item from User A (the provider) by including her/his usage environment description. When content characteristics and usage environment conditions do not match, User B redirects the request including the content Digital Item (multimedia resource + content-related metadata) and the context Digital Item (usage environment description) to the adaptation service that performs the adaptation accordingly. Thereafter, the adaptation service provides the adapted content Digital Item to User B who is now able to consume the adapted Digital Item.

Implementations of such an adaptation service are designated as media gateways or Content Adaptation Nodes (CANs). Within the scope of the IST-DAIDALOS project [24], we implemented such Content Adaptation Nodes utilizing different IETF and MPEG control and description technologies. The CAN operates as a network service and supports a certain set of adaptation operations.
It is registered in a Service Discovery Server (SDS) within a domain and can be located via the Service Location Protocol (SLP) [25]. Each registration comprises the capabilities of the respective CAN, since different adaptation nodes may offer completely different content adaptation services (e.g., supporting different media codecs, different control and transport protocols, etc.). Therefore, a model for describing content adaptation services has been developed.

The service was designed in a flexible and modular fashion so that it can be deployed over a diverse set of network environments and tailored to the specifics of the desired usage scenario. However, in the scope of the DAIDALOS project, the service was deployed on a multimedia provisioning platform based on SIP [5] for session negotiation, on Megaco [7] for loosely-coupled media gateway control and on a combination of SLP [25] and OWL [26] for Service Discovery.

Within each CAN, a decision-taking engine monitors the adaptation process and the feedback about the currently perceived network QoS (e.g., obtained via the RTCP [8]). If necessary, the decision-taking engine may vary the current parameters of the adaptation session. Thus, even environments with highly variable network QoS and nomadic terminals that may roam technologically different networks can be supported. Of course, the decision-taking engine must be informed about the variation boundaries of the adaptation parameters. MPEG-21 DIA descriptions (e.g., of usage environments and user preferences) are the perfect tools for describing these boundaries, but they are not supported via the traditional SDP/SDPng. Our SDPng++ model, on the other hand, allows using such MPEG-21 DIA descriptions, thus improving the utility of CANs.

Regarding usage scenarios, two main approaches are tackled:

- **Implicit Adaptation Services** – The implicit adaptation is triggered and configured user-transparently via the session control elements located in the core network. This type of adaptation service is intended for users equipped with legacy terminals that are not capable of detecting environment changes and of configuring the adaptation process on their own. Figure 5 shows a deployment example of such a service. When the session control plane (the SIP proxy, in this case) receives the content request from the consumer (1) it checks his/her user profile as well as the associated environmental constraints (e.g., network status, policy, provider capabilities, etc.) and decides whether an adaptation service should be inserted in the multimedia session or not (2). Several alternatives were evaluated to make the session control elements aware of provider capabilities, such as including them in the user profile, proxy-driven polling after provider registration, splitting the transactions in several different dialogs using a SIP Application Server behaving as a Back-to-Back User Agent [27], etc. In case adaptation is required, the proxy searches for suitable CANs through Service Discovery and selects the most appropriate element or a combination of them (3-4). The adaptation session is configured in the selected CAN using Megaco (5), and the information on adaptation settings returned in the CAN responses (6) is used by the proxy to modify the initial multimedia session description offer provided by the consumer (7) prior to delivering it to the producer (8). In this particular example, the Megaco transactions would involve the creation of a new context at the CAN and the association to it of a send-only termination for the consumer and a receive-only termination for the producer. The producer includes in the SIP response a multimedia session description containing his/her answer to the received offer (9). The answer also is modified (10) by the proxy using the information obtained in (6) prior to forwarding it to the consumer (11). The response acknowledgement (12-13) is followed by media exchange start (14-15).
• **Explicit Adaptation Services** are triggered and configured directly by user terminals with no aid from the network logic apart from the basic media setup capabilities. This type of adaptation service is intended for users equipped with advanced terminals allowing them to setup and control the adaptation service. Figure 6 shows an example of session setup using this type of adaptation service. After an unsuccessful session establishment due to capabilities mismatch (1-6), the consumer decides to explicitly invoke a content adaptation service in order to be able to receive the desired media (7). After searching and selecting a suitable CAN applying Service Discovery (8-9), the consumer composes the adaptation service description (10) and configure it in the CAN using SIP third-party call control (11-16) [28]. The CAN behaves in this case as a SIP terminating User Agent, in order to allow media gateway configuration through a protocol suitable for end-terminals. The information returned with the CANs response is later on used from the consumer to modify his/her initial offer to the producer (17), leading this time to a successful session establishment (18-23) allowing media exchange through the CAN (24-25). The adaptation process can also be triggered by the producer applying the inverse third-party call control procedure to the shown one.

Both implicit and explicit adaptation services can safely coexist thanks to the application of different user profiles and network policies that enable dynamic selection of adaptation services to be used within the session context.

5. Conclusion

This paper presented a novel approach allowing for convergence between high-level, protocol-independent multimedia description formats and low-level, multimedia-enabled network protocols. The former (defined by MPEG) provide means for the construction of network-enabled device and coding format independent adaptation services and the latter (defined by IETF) are used for transport, negotiation and exchange of these descriptions. Consequently, we introduced a general definition and control model and implementation thereof satisfying the requirements of already existing protocols (i.e., SDP) as well as protocols under development (i.e., SDPng). Seamless integration of media description formats as defined within MPEG-21 DIA is also provided. The scalability of the implementation is shown through tests and the usefulness of our SDPng++ model is demonstrated through the realization of two common usage scenarios within the IST-DAIDALOS project for configuring Content Adaptation Nodes.

Since the SDPng draft has not yet achieved RFC status within the standardization process of the IETF, it should also be considered to integrate our proposals in future versions of SDPng. We also believe that the standardization process of SDPng should be emphasized, since the advantages over the older SDP standard are obvious.

6. Acknowledgements

The work described in this paper is based on results of IST FP6 Integrated Project DAIDALOS, which receives research funding from the European Community's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Additional support has been provided by the DFG within the AKOM framework.

References:


[21] E. Gamma et al., Design Patterns – Elements of Reusable Object-Oriented Software (Addison-Wesley, Reading-Massachusetts (USA), 1994)


[26] Eric Miller et al., Web Ontology Language (OWL), W3C Specification, Feb. 2004


[28] G. Camarillo et al., Transcoding Services Invocation in the Session Initiation Protocol (SIP) Using Third Party Call Control (3pcc), IETF RFC 4117, June 2005