AUTOMATIC ADAPTATION OF STREAMING MULTIMEDIA CONTENT IN A DYNAMIC AND DISTRIBUTED ENVIRONMENT

Andreas Hutter[1]*, Peter Amon*[1]*, Gabriel Panis*[1]*, Eric Delfosse*[2]*, Michael Ransburg*[3]*, Hermann Hellwagner*[3]

[1]Siemens AG, [2]IMEC, [3]Klagenfurt University

ABSTRACT

The diversity of end-terminal and access network capabilities as well as the dynamic nature of wireless connections pose significant challenges to providers of multimedia streaming services. In this paper, we present a system based on MPEG-21 Digital Item Adaptation (DIA) technologies that automatically adapts scalable multimedia resources, like upcoming MPEG-4 Scalable Video Coding (SVC) streams, in a generic and transparent way to the user and session context. This context includes terminal and network capabilities as well as user characteristics. A server side adaptation engine reacts to context changes by dynamic decision taking and accordingly modified bitstream adaptation. Furthermore, novel concepts are presented that facilitate multimedia adaptation in a distributed fashion along the delivery path.

1. INTRODUCTION

Adaptation of multimedia resources is a valuable technique for service providers to: a) target a wide range of devices/networks and consequently users, b) maximize the customers' experience and c) minimize storage and maintenance requirements on the server side. In order to reduce complexity, it is desired that the adaptation decision taking and resource adaptation are automatic and generic and media format independent. Adaptation is especially advantageous in a streaming scenario, which imposes additional requirements such as dynamism and low delay. The technologies presented in this paper build a multimedia resource adaptation architecture, including the decision taking and resource adaptation mechanisms. The approach facilitates the scalability properties of recent media codecs such as the scalable video codec currently under study for standardization in MPEG-4 SVC [2]. Our adaptation engine is based on MPEG-21 DIA technologies [1] that have been extended to support dynamic adaptations. Furthermore, a scenario has been investigated where the adaptation is applied in an intermediate network node instead of, or in addition to, the server. This scenario imposes some additional requirements which will be discussed. The paper is organized as follows. In Section 2 application scenarios are provided for the dynamic and distributed adaptation cases. In Section 3 the overall adaptation architecture is presented, followed by an overview of the dynamic decision taking and resource adaptation technologies in Section 4. In Section 5 our approach for the distributed adaptation scenario is presented, followed by concluding remarks in Section 6.

2. APPLICATION SCENARIOS

2.1. Dynamic Adaptation

Dynamic adaptation is understood to be the adaptation of multimedia resources *during* the streaming session. This does not exclude adaptation at the *beginning* of the session which is simply considered a subset of the dynamic adaptation category. A use case where the need for dynamic adaptation arises is presented in the following. A user spends a lot of time on the road and therefore he receives information and multimedia content on his mobile smart phone. His phone is usually connected to the UMTS network, however since it has also an integrated WLAN module, it can seamlessly switch to the higher bandwidth LAN network once an access point is detected.

Assume a user is sitting in a café accessing his multimedia content over WLAN using the access point provided by the café. The content is adapted and streamed by the adaptation engine and streaming server, respectively. The adaptation process takes into account the capabilities of the mobile device and the WLAN network characteristics. When the user leaves the café, his mobile phone seamlessly switches to the UMTS network. The change in the network characteristics is detected and the adaptation engine quickly modifies the adaptation parameters accordingly to further the bitrate of the streamed resources.

2.2. Distributed Adaptation

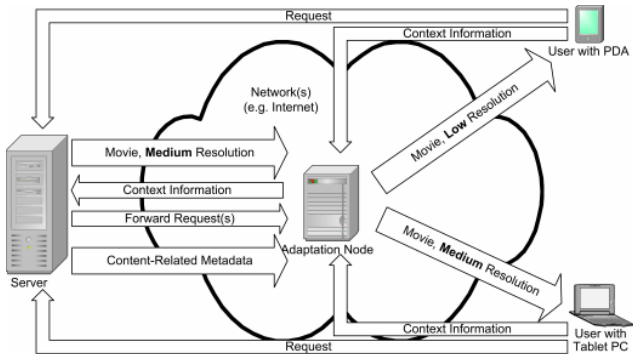

Figure 1: Distributed Adaptation Scenario

Figure 1 illustrates a use case for distributed adaptation: Imagine an Internet-based pay-per-view provider that streams the same movie continuously (e.g., for an entire week) on the same "channel". For a fee, users can join this channel for the duration of the movie. After the payment (which is not addressed here), the user sends a request to join the channel to the server. The server then selects the most appropriate adaptation node (e.g., based on a location description included in the user's request) and forwards the request to the adaptation node. The adaptation node responds to the request and receives context information from the user (e.g., terminal capabilities and current network conditions). It then compares this context information with the context information of all other users which it serves on this channel and decides whether the quality of the stream received from the server is sufficient or not. If not, a higher quality stream is requested from the server. The content is adapted for every user on the adaptation node in a generic way, based on content-related metadata that are streamed alongside the content.

3. ADAPTATION ENGINE ARCHITECTURE

Figure 2 gives an overview of the adaptation engine architecture and its internal (meta-) dataflow, which will be explained step by step in the sequel. The central black-bordered box represents the actual adaptation engine with a generic design such that the same block can be employed on the server as well as on the adaptation node. This is graphically represented by the switch between steps 1 and 2 in Figure 2 (also see Figure 1).

Inside the adaptation engine, the *adaptation engine control* implements the interface to the *streaming server* and *file reader*. Furthermore, as its name indicates, it controls the internals of the adaptation engine by activating the different modules at the appropriate time. Finally it is also responsible for most of the internal communication between the different modules.

Depending on the application scenario the adaptation engine either resides at the server or at a network node. In the former case the content is retrieved from a local repository through the *file reader* that fragments and packetizes the respective media and metadata content to enable streaming (1a). In the latter case the content and its related metadata are retrieved from the network (1b). In both cases the packetized (meta-) data is presented to the depacketizers to extract the respective media and metadata fragments (2a + b).

Part of the depacketized content-related metadata is then presented to the *optimizer* (3), which is responsible for determining the optimum adaptation parameters settings matching the usage environment constraints like, e.g., available bandwidth, display resolution, etc. This module thus performs the adaptation decision taking process referred to earlier.

All the (static) usage environment constraints are stored in a central database by the *context aggregation tool* except the dynamically changing bandwidth, which is provided by the *bandwidth estimation tool*. This tool regularly probes the transmission channel to obtain bandwidth estimates (5a). All this information is then presented to the *optimizer* (4 + 5b), which computes the optimum adaptation parameters.

Next, these parameters are fed into the *resource adapter* (6) that performs the actual adaptation of the resource (8) using the remaining part of the content-related metadata (7). Finally, the adapted media (10) and metadata (9a + b) are packetized and transmitted through the *streaming server* (11a + b).

4. DYNAMIC ADAPTATION

Often the context of a user accessing a multimedia service changes during the session. The most common cases involve fluctuations of the network bandwidth, especially in mobile networks, or the roaming scenario described in Section 2.1. It is essential that the adaptation engine can dynamically react to changing context information to generate a new adaptation decision and apply this decision to the remainder of the resource to be streamed. As mentioned earlier all static context information can be dealt with as a sub-case of the dynamic case. In the next two sub-sections, technologies to enable dynamic decision taking and resource adaptation are presented.

4.1. Dynamic Decision Taking

As described above, the optimizer is responsible for determining the optimum adaptation parameters given some usage environment constraints. This process requires at least the knowledge of the adaptation possibilities, preferably complemented by the resource and quality characteristics that will result from a certain adaptation choice. On the other hand it requires a set of context descriptor values. In the case of a scalable video stream like MPEG-4 SVC the adaptation options would cover all supported combinations of spatial, temporal and SNR resolution resulting in a certain bit rate, while the characteristics of the adapted streams could be described by the resulting average PSNR. The values of the context descriptors would e.g. correspond to the screen size or the current network connection. MPEG-21 DIA specifies two descriptors for storing such information: AdaptationQoS (AQoS) and Universal Constraints Descriptor (UCD) [1]. AQoS describes the relationship between

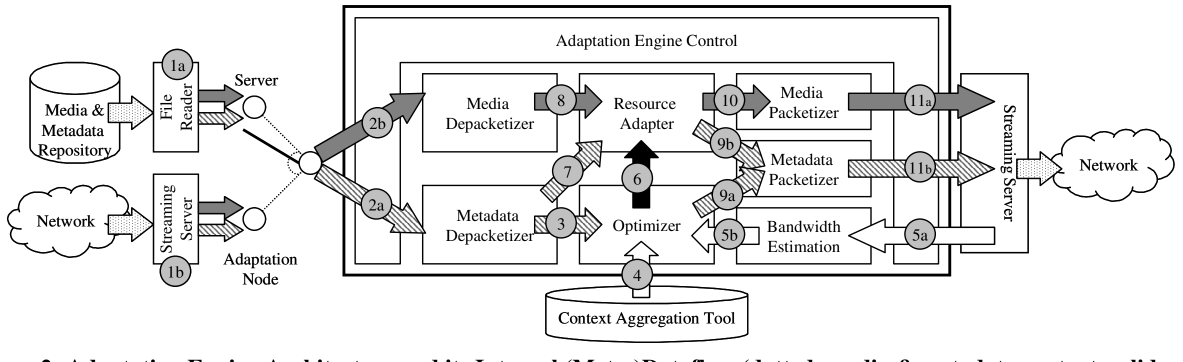

Figure 2: Adaptation Engine Architecture and its Internal (Meta-)Dataflow (dotted: media & metadata content, solid gray: media content, hatched: content-related metadata, white: context metadata, solid black: adaptation parameters)

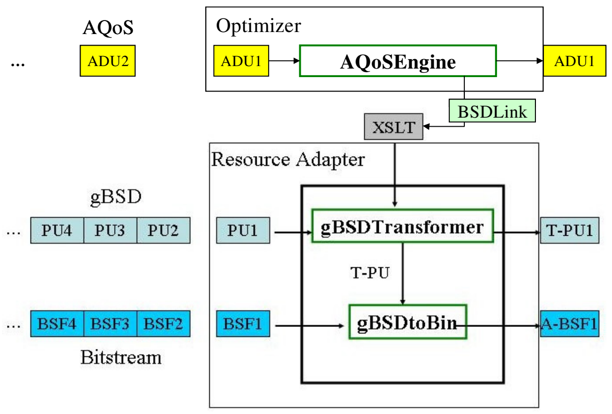

Figure 3: Dynamic Decision Taking and Resource Adaptation

adaptation parameters, resulting resource characteristics, quality and possibly other parameters. The UCD may complement this information by mathematically specifying constraints imposed by the usage environment context or other sources. Furthermore, an optimization function can be declared in the UCD to control the selection of the best adaptation option. Further information on the functionality of these descriptors in the decision taking process can be found in [3].

The problem of adaptation decision taking in a dynamic streaming environment is twofold: (1) the actual resource characteristics, e.g. bitrate, and (2) the context, e.g. bandwidth, may vary a lot between bitstream fragments during streaming. To cope with this the AQoS can be fragmented into so-called *Adaptation Units (ADUs)* using its switching mechanism [1] [3]. An ADU describes the adaptation options and resulting characteristics of one or more successive bitstream fragments (BSFs). Each BSF may correspond to a number of GOPs in the case of SVC. Each ADU can be transmitted and processed independently of one another. This is shown in the upper part of Figure 3, where ADU1 contains adaptation information of BSF 1 and 2, and ADU2 of BSF 3 and 4, etc.

Selecting the granularity of an ADU is arbitrary and mainly triggered by trading-off the accuracy with the verbosity of the AQoS description. Indeed, a small number of ADUs, each describing the average characteristics of a large sequence of BSFs, results in a small AQoS description, but which might be imprecise with respect to the actual characteristic, e.g. bitrate, of each adapted BSF. Hence, the optimizer might produce poor adaptation decisions. A large number of ADUs, on the other hand, each describing a small sequence of BSFs, will result in a larger but more precise AQoS description and therefore better adaptation decisions.

4.2. Dynamic Resource Adaptation

When adaptation decisions have been obtained, another MPEG-21 DIA descriptor, *BSDLink*, specifies the link between the output of the optimizer and the input of the resource adapter

The resource adapter for scalable resources is based on the *generic Bitstream Syntax Description (gBSD)* concepts specified in the MPEG-21 DIA standard [1]. In brief, the adaptation process is abstracted by the use of a high level XML description of the resource's bitstream syntax, the gBSD. By first adapting the gBSD and then processing the modified gBSD to generate the adapted bitstream, the resource adaptation engine is abstracted from the bitstream syntax specifics [5]. In the case of SVC the gBSD will e.g. provide references to independent pieces of the bit stream that need to be dropped when reducing the video format from e.g. CIF to QCIF. The basic concept described in [1] has been extended to better support dynamic adaptations in a streaming scenario. Instead of transforming and processing the entire gBSD at once, the description is fragmented into smaller descriptions called *Process Units (PUs)* [5]. The fragmentation itself is arbitrary, the only requirement imposed is that a PU can be transformed and processed independently of other PUs i.e. there should be no dependencies between PUs for their transformation and processing. An example for the corresponding syntax structure in SVC could be a GOP.

The lower part of Figure 3 illustrates the dynamic resource adaptation architecture. The gBSD has been fragmented into process units (PU1, PU2, …), each describing a fragment of the bitstream (BSF1, BSF2, …). The resource adapter begins processing the PUs and corresponding BSFs sequentially. First each PU is transformed using an XSLT stylesheet that applies all the modifications on the bitstream fragment. The transformed PU (T-PU) is then input to the *gBSDtoBin* process [1] along with the bitstream fragment. The gBSDtoBin processes the PU to generate the adapted bitstream fragment. Then the T-PU and the adapted bitstream fragment (A-BSF) are ready to be streamed. It should be noted that for simplicity the BSDLink that provides the link between the optimizer and resource adapter related descriptors has been omitted in this description of the process. Applying the adaptation piece-wise has several advantages including reduction in memory requirements, low initial delay, applicability of the adaptation process in a distributed scenario, as well as a facility to dynamically change the adaptation parameters.

5. DISTRIBUTED ADAPTATION

In multimedia resource streaming scenarios supporting the adaptation on intermediate network nodes is essential to enable scenarios as the one depicted in Figure 1. This requires, however, the transmission of media data as well as its associated metadata to the adaptation node. In this section, we will illustrate the architecture of the adaptation node and introduce the mechanisms used for the transport of the metadata.

5.1. Adaptation Node Architecture

The architecture of the adaptation node can be coarsely divided into three modules (Figure 4). The *context aggregation module* is responsible for collecting context information from client nodes located downstream on the delivery path, and for providing this information to the other modules. Based on this context information, the *adaptation engine* can decide which adaptation it should apply to the multimedia content that is to be transmitted to the clients (cf. Section 3). The context information is also processed by the *context merging module*, which compiles a set of context descriptors each of which relates to the highest quality which *any* of the connected terminals can consume. The merged context information is enriched by adaptation-node specific descriptors, describing for example the supported adaptations and sent to the server.

Based on the merged context information, the server can adapt the content to the quality level requested by the adaptation node. The adapted content corresponds to the highest media quality which satisfies the requests of *all* terminals connected to the adaptation node. The server then delivers this content together with its content-related metadata, i.e., the BSDLink, the gBSD and the associated transformation style sheet, as well as the AQoS and UCD descriptions [1], to the adaptation node.

Subsequently, the adaptation node replicates and provides the content to each connected terminal in an optimum way. To that end, the adaptation node further adapts and delivers each of the content streams according to the individual context descriptions received from the terminals. Thus, each client can be supplied with the highest quality stream that is possible under the constraints of its specific network, user, and terminal context.

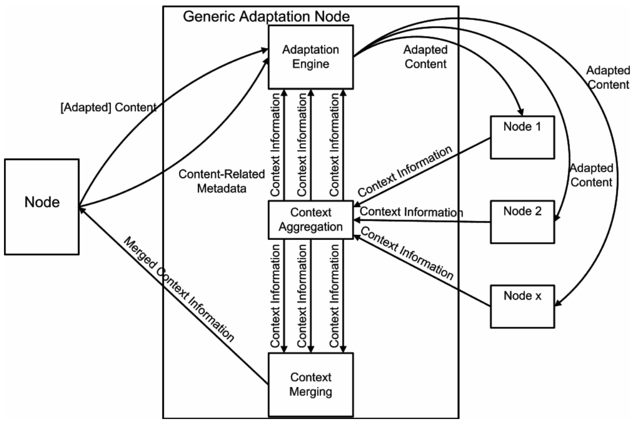

Figure 4: Adaptation Node Architecture

5.2. Adaptation Metadata Transport

Each of the descriptions introduced in Section 4 needs to be transported from the content provider or content consumer to the adaptation engine. Different transport formats and protocols are being used for that purpose.

The context information is transmitted independently from and not synchronized with the media content and the content-related metadata. Therefore, one of the means to transport XML information, such as the HyperText Transfer Protocol (HTTP) or the Simple Object Access Protocol (SOAP), can be employed. In our approach, HTTP is used for the transport of the context information, and the MPEG-21 DIA Configuration descriptor [1] is used to negotiate the context information set.

In general, the BSDLink and the gBSD Transformation need to be available at the adaptation engine before the streaming of the media content (and associated metadata) starts. Therefore, they are transported within the Session Description Protocol (SDP) during the content negotiation phase, using an attribute parameter. Since the gBSD and AQoS are associated with a streaming resource – the media content – that is usually transported by means of the Real-time Transport Protocol (RTP), the gBSD and AQoS are transported using RTP as well. Hence, these descriptions are fragmented for the transport using the concepts of ADUs and PUs (cf. Section 4) and synchronized with the resource. On the stream level, there are three options of how to transport this type of metadata. One can either multiplex the metadata into the media content stream, use one single metadata stream, or use a separate metadata stream for each kind of metadata. Evaluations show that the most feasible way is to transport each metadata stream in a separate RTP stream. A detailed description and evaluation of the mechanisms needed to enable the transport and processing of this type of metadata is provided in [6] including further considerations regarding the fragmentation of the metadata and the synchronization with the media data, both on the transport and the processing levels,

6. SUMMARY

In this paper, we presented a generic architecture of a multimedia adaptation engine supporting dynamic decision taking and resource adaptation. It is based on MPEG-21 DIA descriptors and forms a generic adpatation framework that can support adaptation across codecs and media types, such as. the upcoming MPEG-4 SVC The MPEG-21 DIA architecture was extended to additionally support dynamic decision taking and resource adaptation. Furthermore, another extension for the scenario where the adaptation is distributed over the server and intermediate network nodes, was also discussed.

7. ACKNOWLEDGEMENT

Part of this work was supported by the European Commission in the context of the DANAE project (IST-1-507113) [7].

8. REFERENCES

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

[2] H. Schwarz, D. Marpe, T. Wiegand, "MCTF and Scalability extension of H264/AVC", *Picture Coding Symposium (PCS) 2004*, December 2004

[3] D. Mukherjee, E. Delfosse, J.G. Kim and Y. Wang, "Optimal Adaptation Decision-Taking for Terminal and Network Quality of Service," to appear in *IEEE Transactions on Multimedia*, vol. 7, no. 2, April 2005.

[4] C. Timmerer, G. Panis, and E. Delfosse, "Piece-wise Multimedia Content Adaptation in Streaming and Constrained Environments (invited paper)," *Proceedings of the Sixth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2005).*

[5] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, "Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments," *to appear in IEEE Transactions on Multimedia,* vol. 7, no. 2, April 2005.

[6] M. Ransburg, C. Timmerer, and H. Hellwagner, „Transport Mechanisms for Metadata-driven Distributed Multimedia Adaptation," *Proceedings of the First International Conference on Multimedia Services Access Networks (MSAN'2005).*

[7] DANAE Website: http://danae.rd.francetelecom.com