A COMPARATIVE STUDY OF STATE-OF-THE-ART ADAPTATION TECHNIQUES FOR SCALABLE MULTIMEDIA STREAMS

Andreas Schorr, Franz J. Hauck
University of Ulm
89069 Ulm, Germany

Bernhard Feiten, Ingo Wolf
T-Systems Enterprise Services GmbH
Berlin, Germany

Keywords: Adaptation, scalable media formats, media filtering, MPEG-21, digital item adaptation.

Abstract: Stream adaptation is a key technology for enabling communication between heterogeneous multimedia devices and applications, possibly located in heterogeneous wired or wireless networks. Converting already compressed multimedia streams into a format suitable for a certain receiver terminal and network can be achieved by transcoding or by filtering of media streams. Transcoding allows more flexible adaptation operations but is in general a very CPU-intensive process. Therefore, scalable media formats have been developed, which allow more efficient adaptation of media streams through media filtering. Several filter techniques for specific media formats have been proposed and implemented during the last decade. Recently, the MPEG-21 Digital Item Adaptation standard has defined a set of new tools for multimedia adaptation. In this paper, we provide a comparative study of several adaptation techniques for scalable multimedia streams. We compare generic MPEG-21-based adaptation techniques with filter mechanisms for specific media formats with respect to the required processing resources and scalability. We also compare filter techniques with stream adaptation through transcoding. Moreover, we compare how adaptation of multiple media streams performs on systems with single-core and with multi-core processors.

1 INTRODUCTION

During the last two decades, enormous advances have been achieved in the area of multimedia data compression as well as in (wireless) network technologies. Accompanied by a clear trend towards all-over-IP multimedia communication, these advances enable real-time multimedia streaming applications, such as voice-over-IP (VoIP), video-on-demand (VoD) or video-conferencing (VC). On the other hand, there exist many different devices with different hardware and software capabilities as well as a variety of heterogeneous networks with different properties. If communicating applications do not support the same media formats, or whenever the datarate of a media stream does not match the bandwidth constraints of a certain network, adaptation can be used as a means to bridge the heterogeneity gap.

In this paper, we focus on media adaptation techniques that can be performed in real-time during an ongoing streaming session from the media source to one or several receivers. Such an adaptation can be performed at the source or at the sink of a media stream or somewhere in between (Kassler and Schorr, 2003), and it can be achieved by transcoding or by filtering of media streams or by a number of other techniques that we will shortly discuss in Chapter 2. Transcoding allows flexible adaptation operations but is in general a very CPU-intensive process. Therefore, scalable media formats have been developed, which allow more efficient adaptation of media streams through media filtering, e.g., MPEG-2 (ISO/IEC, 1994), MPEG-4 (ISO/IEC, 2000), WaveVideo (Fankhauser et al., 1999). Several filter techniques for specific media formats have been proposed and implemented during the last decade (Yeadon et al., 1996b; Yeadon et al., 1996a; Keller et al., 2000; Kassler et al., 2001). Recently, the MPEG-21 Digital Item Adaptation (DIA) standard (ISO/IEC, 2004) has defined a set of new tools for multimedia adaptation. These tools make it possible to build generic adaptation engines which can handle arbitrary media formats.

Several researchers have proposed architectures for video gateways (Amir et al., 1995) or content adaptation nodes (Kassler and Schorr, 2003), which are located in the network between sender and receiver(s) of a media stream and provide stream adaptation ser-

Schorr A., J. Hauck F., Feiten B. and Wolf I. (2006).
A COMPARATIVE STUDY OF STATE-OF-THE-ART ADAPTATION TECHNIQUES FOR SCALABLE MULTIMEDIA STREAMS.
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, pages 81-88
DOI: 10.5220/0001568200810088

vices. Such adaptation nodes must be able to adapt large numbers of streams in parallel for being profitable. Whereas commercially distributed media gateways already support real-time transcoding of limited numbers of audio streams, little effort has been spent so far on integrating media filter techniques for scalable media streams into such products. An important question is how well filter techniques perform with respect to required processing resources. In Chapter 3, we compare several state-of-the-art adaptation mechanisms. We compare MPEG-21 DIA based adaptation techniques with media filters for specific media formats and with transcoding techniques. Whereas other performance studies usually concentrate on the quality degradation caused by adaptation operations, our study analyses the scalability of the adapatation algorithms, i.e., how many media streams can be processed on an adaptation node in parallel. Vendors and operators of media gateways might be particularly interested in how currently emerging MPEG-21 DIA based adaptation techniques perform in comparison to other filter and transcoding mechanisms. We also compare how adaptation of multiple media streams performs on systems with single-core and with multi-core processors.

2 ADAPTATION TECHNIQUES

The main focus of our performance study lies on media filter techniques for scalable media formats that can be applied on an intermediary network node between the sender and the receiver(s) of a media stream. Nevertheless, there exist a number of other adaptation techniques for multimedia sessions. These are shortly discussed in Section 2.1. Transcoding and filter techniques are introduced in Section 2.2 and in Section 2.3.

2.1 Sender and Receiver Based Adaptation

Some VoD servers store multiple representations of the same movie clip in their database using different media formats or data rates. When adaptation of an ongoing streaming session is required, *stream switching* can be applied to adapt the multimedia stream (Amon and Pandel, 2003). A drawback of this sender-driven adaptation technique is that more storage capacity is needed for each additional pre-encoded stream representation stored on the video server. Therefore, the number of different formats and data rates that can be offered is limited. For broadcast services, multiple versions of the same multimedia content can be transmitted simultaneously in several separate broadcast channels using different media formats or data rates. This technique is called *simulcast* (Amon and Pandel, 2003). Receiver-driven adaptation of such a streaming session can be achieved by simply switching to another broadcast channel. The drawback of this approach is that a lot of network bandwidth is required (and possibly wasted) if many different media formats shall be supported. A more efficient approach is *Receiver-Driven Layered Multicast (RLM)* (McCanne et al., 1997). Here, different layers of a scalable media source are streamed to different multicast addresses, and receivers decide about the quality by joining or leaving respective multicast groups. The bandwidth requirements for the RLM approach are lower than for simulcast, but if the number of layers is high, the overhead caused by packet headers becomes worse. Newer scalable media formats like fine granularity scalability (FGS) (ISO/IEC, 2000) do not create a discrete number of layers proportional to the number of quality levels. Instead, an FGS-encoded bitstream can be truncated at any arbitrary byte position. This kind of scalability can not be supported by the RLM approach; media filters (see Section 2.3) are better suitable for reducing the data rate of an FGS-encoded media stream.

The above mentioned adaptation techniques have their raison d'être in certain application scenarios, but content adaptation nodes inside the network must offer complementary media filtering or transcoding services if arbitrary terminals, networks and users with their respective requirements on media formats and data rates shall be supported. Nevertheless, it should be mentioned that stream switching, simulcast and RLM all perform excellently with respect to processing resources (neglecting bandwidth and storage resources). In the case of stream switching and simulcast, no additional processing resources are required for adaptation. Using RLM, only a small amount of additional processing power is required for managing multiple parallel multicast sessions and reassembly of the media stream.

2.2 Transcoding

Transcoding denotes the process of changing the format of a media stream by decoding and re-encoding. Input format and target format may both be compressed formats, or either of them may be an uncompressed format. The codec of the input format may either be the same as the target codec, or it may be a different one. In case input codec and target codec are the same, other properties of the media format are changed (e.g., frame size, data rate). A simple video transcoder, for instance, works as follows: a compressed frame is decoded; intermediate operations are applied on the uncompressed data (e.g., scaling the frame size); the video frame is re-encoded. In the lit-

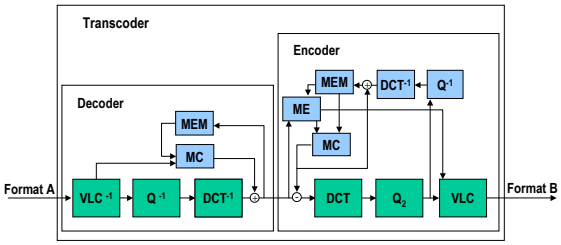

Figure 1: A spatial domain cascaded transcoder.

erature, this kind of transcoder is called a *spatial domain cascaded transcoder* (SDCT) (Sun et al., 2005). Although this approach is very straightforward, the computational complexity of such a transcoder can be very high, especially for video transcoding.

The majority of current video codecs use a block-based video coding scheme, which applies discrete cosine transform (DCT) on a block of video pixels, quantizes the DCT coefficients followed by variable length coding (VLC) of the quantized values. Furthermore, motion-compensated predictive coding is often used for reducing the temporal redundancies between video frames. Thus, the main processing steps inside an SDCT (as depicted in Figure 1) are: inverse VLC, inverse quantization, inverse DCT and motion compensation (MC), followed by DCT, quantization (Q), and finally VLC. Furthermore, the re-encoding step requires additional Q^{-1} and DCT^{-1} operations to allow proper motion estimation (ME) and motion compensation.

Several optimized transcoding techniques have been proposed that allow faster transcoding. For instance, motion vectors of the input stream can often be re-used for the output stream. Thus, it is not necessary to perform new motion estimation. If the spatial resolution of the picture shall be reduced, motion vectors must to be down-scaled, which can still be performed much faster than a new motion estimation search. Other optimization techniques perform conversion operations in the frequency domain instead of transforming the video data into the spatial domain first. This way, DCT and DCT^{-1} operations can be skipped. Vetro et al. provide a quite comprehensive survey of such optimized transcoding techniques (Vetro et al., 2003).

Sun et al. declare that optimized video transcoders achieve temporal performance gains of up to 70 percent over simple spatial domain cascaded transcoders (Sun et al., 2005) (possibly at the cost of reduced video quality). Even when using such an optimized video transcoder, the computational complexity is high in comparison to filter techniques, which are discussed in the next section. A few video codecs exist which use wavelet transform instead of block-based DCT (Fankhauser et al., 1999), but the computational complexity of these compression algorithms is similar to the complexity of codecs using DCT. Many audio codecs require substantially less processing resources than video codecs, but the requirements for more complex audio codecs like MP3 or AAC are also quite high in comparison to filter techniques.

2.3 Media Filtering

Media filtering denotes the process of removing information from a media stream. Filtering media data compressed with a conventional coding format is difficult and may require costly decoding and encoding operations. Scalable media formats like MPEG-4 (ISO/IEC, 2000), or WaveVideo (Fankhauser et al., 1999) have been developed which allow more efficient filtering without decoding the media stream. Today, two techniques are common for creating scalable media formats: *layered coding* and *fine granularity scalability coding* (FGS). In layered coding schemes, frame data is spread over several media layers. The first layer is called *base layer*, the others are called *enhancement layers*. The base layer is always decodable without the enhancement layers, but the quality of the media stream will be low if only the base layer is used for decoding. The more enhancement layers are available at the decoder, the better the quality of the media stream will be. In the FGS encoding scheme, the media stream is also split into a base layer and an enhancement layer. Additionally, the information in the enhancement layer can be truncated at any arbitrary byte position while still being decodable.

A media filter removes certain enhancement layers from a layered bit stream or truncates an FGS encoded bit stream, thus reducing the data rate of the stream. Data rate reduction can also be achieved by transcoding, but the computational complexity of a simple filter operation is much lower. A filter only has to identify where the data belonging to each layer is located inside the bitstream and to remove some of the identified parts. *SNR (signal-to-noise ratio) filters* allow to scale the quantization accuracy. For video data, there exist also *spatial* and *temporal filters* that allow to scale the spatial and temporal resolution of the video. Furthermore, some scalable video formats like WaveVideo encode luminance and chrominance pixel information in different layers, thus allowing to reduce the SNR of luminance and chrominance independently or to remove the chrominance information completely (Kassler et al., 2001).

MPEG-21 part 7 also known as *Digital Item Adaptation* introduces a novel approach for the adaptation of media resources. As a basis, the model of a *Digital Item* is chosen, which is composed of structured metadata and binary resources. The XML based metadata is normatively defined using XML Schema and allows to combine metadata describing the resources itself

as well as metadata describing user preferences, usage environment, usage constraints, quality of service information and other descriptions with the media resources. A generic adaptation model is specified, that applies a normative processor as part of a digital item adaptation engine. Following this model, it is possible to adapt media encoded with scalable codecs without the need of codec specific processors. The idea behind that model is the introduction of a management layer fully based on XML. A certain number of descriptions is needed to manage the resource adaptation, one of them is the so-called *Bitstream Syntax Description* (BSD). In principle, this BSD describes a media resource as a sequence of units and can be transformed using e.g., XSLT (other transformation languages like STX are possible as well). The transformed BSD is then fed into the normative *BSDto-Bin processor*, which applies the changes to the media resource. This normative processor applies only simple operations like truncation or update controlled by the BSD. The transformation logic is placed into the transformation stylesheet, that becomes part of the metadata of a digital item. Nevertheless, a BSD-based adaptation engine still needs to know a codec specific bitstream syntax schema. Therefore, MPEG-21 DIA also defines the so-called *generic BSD (gBSD)*, which can be used to describe any binary resource in a codec independent manner and no codec specific schema is required for adapting the bitstream.

3 MEASUREMENTS

In next generation networks, content adaptation nodes (CANs) belonging to the network infrastructure will offer adaptation services for multimedia streams. The purpose of the following performance measurements is twofold. If real-time media streams shall be adapted on such a CAN inside the network, it is important to know how much delay is added to the end-to-end latency of the data transport. Furthermore, we study the scalability of adaptation services. We are interested in how many streams can be adapted in parallel on a single CAN. In the latter context, we also look at the performance gains that can be achieved by building such an adaptation service with multi-processor or multi-core processor technology.

3.1 Test Environment

The examined adaptation mechanisms were implemented as software plug-ins for a generic content adaptation node and tested on two different hardware platforms. Test system 1 (TS1) used the following hardware configuration: Pentium-4 processor with 1,8 MHz clock rate, 1 GB RAM, 100 Mbit/s network interface. Test system 2 (TS2) was configured as follows: Athlon-64 X2 Dual-Core 3800+ processor, 1 GB RAM, 100 Mbit/s network interface.

We examined codec-specific filter mechanisms for MPEG-4 BSAC audio and WaveVideo streams, a gBSD-based MPEG-21 DIA filter for audio streams, and video and audio transcoders. The gBSD-based adaptation engine was implemented using the C/C++ based Expat XML parser library (libexpat) and the Sablotron XSL transformation library (sablot) — both are developed in open source projects. As a design decision, the event based Simple API for XML (SAX) was chosen due to performance advantages over the more complex Document Object Model (DOM) API. The examined filter mechanisms for WaveVideo were proposed by (Kassler et al., 2001). Transcoding is an adaptation operation not explicitly dealing with scalable media streams. Nevertheless, transcoding can of course be applied on scalable and on non-scalable media streams alike, so it is interesting to know whether filter mechanisms perform much better than transcoding or not. We evaluated a proprietary cascaded audio transcoder and a proprietary spatial domain cascaded video transcoder. For encoding and decoding, the cascaded audio transcoder makes use of standard audio codecs installed on the Windows XP platform as well as of the well-known LAME MP3 encoder. The cascaded video transcoder uses the well-known XVID codec for MPEG-4 encoding.

3.2 Measurement Results

In this section, we present the results of our performance measurements. The following diagrams depict the processing time (measured in microseconds) per audio/video frame. Each diagram (except Figure 3 and 7) shows four measurements: two curves depict the processing time for a single media stream on TS1 and TS2, respectively. The third and fourth curve show the total processing time for parallel adaptation of multiple media streams on TS1 and TS2, respectively. In all four cases, the same adaptation operations are applied on each media stream, and in each test sequence, the adaptation parameters are changed over time as depicted in the respective diagram. Please note that for better readability of the diagrams the depicted processing time values are smoothed average values (over the last ten frames).

Note also that the measurement includes only the processing time inside the content adaptation node. For determining the total delay added to the end-to-end latency, you have to add the time required for receiving and sending the data through the CAN's network interface and the time for which the incoming data is stored inside the CAN's jitter buffer. The time needed for receiving and sending can be easily calculated for constant bitrate media streams. For instance,

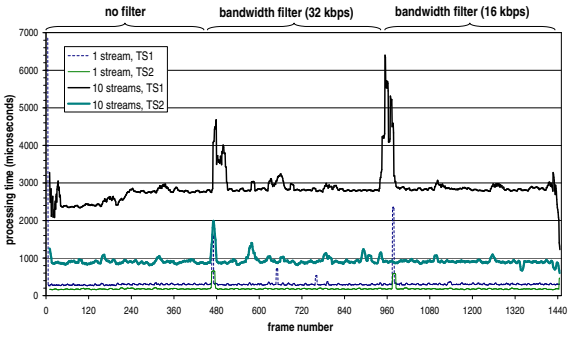

Figure 2: Simple filter for MPEG-4 BSAC audio.

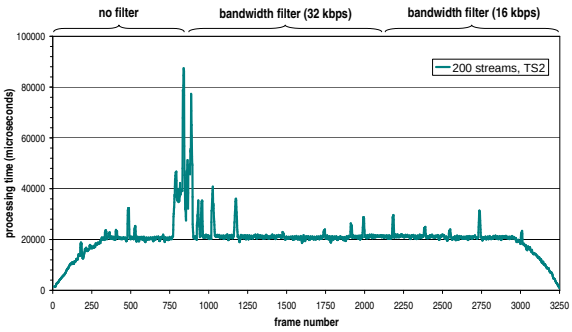

Figure 3: Maximum number of adaptation sessions.

when processing a 64 kBit/s audio stream, the time required for sending and receiving a frame of 40 milliseconds length through a 100 MBit/s network interface is (64 kBit/s / 25 frames/s) / 100 MBit/s = 25,6 μs. The time for which the incoming data packets are stored in the jitter buffer is variable. Setting it to zero will result in loss of information if the network jitter is more than zero and temporal dependencies between consecutive frames exist, because adaptation modules like transcoders must processes such media units in the correct temporal order. So you usually have to set the length of the jitter buffer to a value corresponding to the currently monitored network jitter.

3.2.1 Test Sequence 1: Codec-specific Audio Filter

In a first test sequence, we evaluate the performance of a proprietary codec-specific media filter for MPEG-4 BSAC audio streams (Figure 2). The original stream has a data rate of 64 kBit/s. The test sequence has a length of 45 seconds. During the first 15 seconds, the stream is not modified by the CAN. During the next 15 seconds, the media filter reduces the data rate of the audio stream to 32 kBit/s. During the third period, the data rate is reduced to 16 kBit/s. The average processing time for a single audio frame (40 ms of audio data) is quite low (290 μs on TS1 and 175 μs on TS2). Since the filter operation is quite simple, there is no visible difference (in processing time) during the first period, where no adaptation was performed, when compared to the second and third period. You can also see that each time the filter settings are changed, a short peek occurs which is due to the fact that the filter has to be re-initialised. A few additional peeks occur randomly, which are probably caused by interrupt routines executed by the operating system.

Moreover, Figure 2 shows that the audio filter has good scalability properties. We can see that processing time is linearly increasing with the number of audio streams. For instance, filtering ten audio streams requires 2.8 milliseconds total processing time on TS1, which is about ten times more than the processing time per audio frame for a single stream on TS1. On TS2, processing ten audio streams requires about 950 μs, which is 5.5 times the processing time for a single audio stream on TS2. The performance gain achieved by using a dual-core processor is $10/5.5-1 = 81\%$. This performance gain is less than 100 percent, because the total processing time inside the CAN is not only determined by the CPU, but also by other hardware components, such as main memory and PCI bus.

Just for the purpose of demonstrating the maximum number of adaptation sessions that can be executed in parallel, Figure 3 shows an additional measurement with 200 adaptation session on TS2. The average total processing time per frame is now about 20 ms, which is 114 times the processing time per audio frame for a single stream on TS2. Assuming that a single-core system would require 200 times the processing time for a single stream, the performance gain through the dual-core technologie is $200/114-1 = 75\%$. In Figure 3, a relatively long and high peek occurs when the filter is activated because this is done for 200 sessions at the same time. In a real-life scenario, it is very unlikely that this happens at the same time for all adaptation sessions. Furthermore, we can see that the peek is not always so high. At the transition from period two to period three, there is almost no peak visible.

3.2.2 Test Sequence 2: gBSD-based Audio Filter

In the second test sequence, the same audio stream is adapted in the same way as in test sequence one. But this time, a generic MPEG-21 DIA adaptation engine performs the adaptation operation using gBSD metadata that is transmitted together with the audio stream. The result is depicted in Figure 4.

It is not possible to filter more than three streams in realtime on TS1 because of the high CPU demands of the gBSD engine. Therefore, we cannot directly compare the processing times for ten parallel ses-

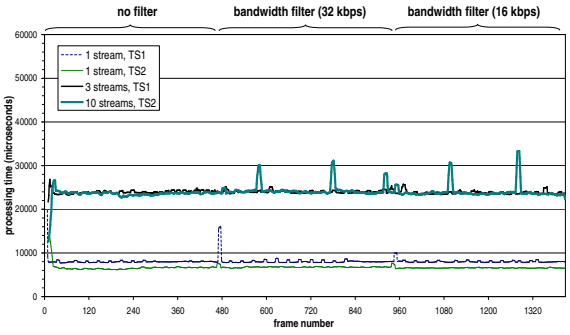

Figure 4: gBSD filter for MPEG-4 BSAC audio.

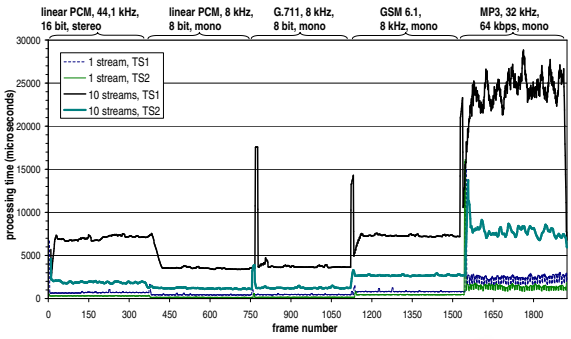

Figure 5: Audio Transcoding.

sions on TS1 and TS2. The processing time needed by this gBSD-based adaptation engine is obviously much higher than the processing time needed by the codec-specific filter used in test sequence 1. Actually, about 90% of the processing time is required for the XML transformation, whereas the actual filter operation is quite simple and fast. Furthermore, an obvious weakness of the current implementation is that the gBSD processing (including XSL transformation) is performed even when no filtering is necessary. A future version of this implementation may be enhanced with a fast test at the beginning of the filter operation to determine whether filtering is necessary at all, thus reducing the processing time during periods where no adaptation is required.

Figure 4 shows as well the performance gain achieved by using a dual-core CPU. Whereas filtering a single stream on TS2 requires about 6 ms per frame, the total processing time for ten parallel streams is only about 24 ms, which is four times the processing time for a single stream on TS2. The processing time for three streams on TS1 (also about 24 ms) is exactly three times more than the processing time for one stream on TS1 (about 8 ms). Assuming that a single-core system with enough processing power would require ten times the processing time for a single stream, a performance gain of $10/4 - 1 = 150\%$ is achieved. A performance gain of more than 100% may occur if the activities of one processor core lead to a higher cache hit ratio for the other core. The Athlon X2 uses the MOESI cache coherency protocol (Sweazey and Smith, 1986) which allows a processor core to access modifications of shared data in the other core's level two cache without accessing the main memory. The hard disk cache may be another source of the improved performance. Actually, we observed that much more paging activity occurs when applying the gBSD adaptation engine as compared to the other examined adaptation mechanisms.

3.2.3 Test Sequence 3: Audio Transcoding

In a third test sequence, we study the performance of audio transcoding performed on a content adaptation node. Here, the CAN receives an audio stream encoded with linear pulse code modulation (PCM) at 44.1 kHz, 16 bit per sample, stereo quality. During the first measurement period, no adaptation is performed. During the second period, sampling rate and quantization resolution are reduced. Then we transcode the stream into G.711, GSM 6.1 and MP3 format, respectively. The required processing time per audio frame is depicted in Figure 5. Notice that the processing time in the second and third period is even lower than the processing time in the first period, during which no adaptation is performed. This is due to the fact, that the total size of the audio data is reduced when reducing the sampling rate, so subsequent operations on this small amount of audio data are performed faster.

An interesting observation is that the transcoding time for all target formats is lower than the processing time needed by the MPEG-21 DIA filter that was examined in test sequence two. This is a bit disappointing, since one of the original goals of MPEG-21 DIA was to achieve very fast adaptation through simple and efficient editing-style operations (ISO/IEC, 2004). However, it can be expected that a higher benefit might be achieved when video scaling based on MPEG-21 DIA is applied.

Since we had no BSAC encoder plug-in for the content adaptation node, we cannot directly compare measurements for BSAC transcoding and BSAC filtering. BSAC encoding requires even more processing resources than encoding MP3. BSAC is based on the MPEG AAC codec (Advanced Audio Coding). Only the stage in the codec that encodes and decodes the already quantized spectral coefficent is different. The BSAC uses arithmetic coding in a way that the coeffiecients are represented in a sliced form. By this means the bitrate can be adapted by removing the slices partly. The transcoding requires cascaded AAC

decoder and encoder. The AAC encoder is much more complex than the decoder, because the encoder usually contains a powerful spectral analysis to adapt the quantisation noise to the auditory perception. Scaling with BSAC does not need complete audio decoding and re-encoding. Only the bit sliced arithmetic decoding has to be processed to get the exact boundaries for removing parts of the bitstream. This arithmethic decoding makes up about 10% of the total decoding process. From these facts, one can estimate the advantage of filtering (with processing gBSD) over transcoding. The AAC and the BSAC decoder have similar complexity. The AAC encoder has at least a complexity that is twice the decoder complexity. Consequently the transoding requires more than 30 times of the processing power than the scaling of the bitrate.

Regarding scalability of the transcoding operations, we can see that the processing time is again linearly increasing with the number of audio streams. For instance, during the fourth period when transcoding to GSM 6.1 is applied, the processing time for a single stream is about 800 μs on TS1 and about 450 μs on TS2. Transcoding 10 audio streams into the GSM 6.1 format requires about 7.5 ms (9.5 times more) processing time on TS1 and about 2.5 ms (5.5 times more) on TS2. We achieve a performance gain of $9.5/5.5 - 1 = 72\%$ by using the dual-core technology.

3.2.4 Test Sequence 4: Codec-specific Video Filter

In a fourth test sequence, we analysed the performance of a media filter for WaveVideo streams (Kassler et al., 2001). This filter is again a codec-dependent filter not using MPEG-21 DIA mechanisms. The results of this test sequence are depicted in Figure 6. Again, no adaptation is performed during the first period of the test sequence. During the second period, the chrominance information is removed from the stream, and during the third period the data rate (originally $>$ 1 MBit/s) is reduced to 200 kBit/s.

As in test sequence one, the filter itself does not require noteworthy processing resources. Simply forwarding the stream requires about 1 ms per frame (because of the high data rate), but when adaptation is actually switched on, the processing time does not increase. Instead, it decreases because of the decreased frame size. Again, the total processing time is increasing linearly with the number of parallel sessions. For instance, during the second period when applying the screyscale filter, the processing time for a single stream on TS1 is about 1.25 ms on TS1 and about 1 ms on TS2. Filtering ten video streams in parallel requires about 14 ms (11 times more) processing time on TS1 and about 8 ms (8 times more) on TS2. The performance gain through the dual-core technology is $11/8 - 1 = 38\%$.

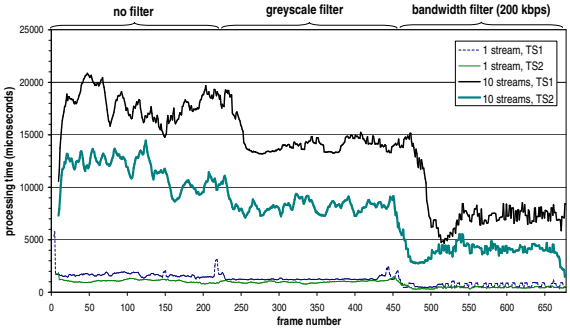

Figure 6: Wave Video Filter.

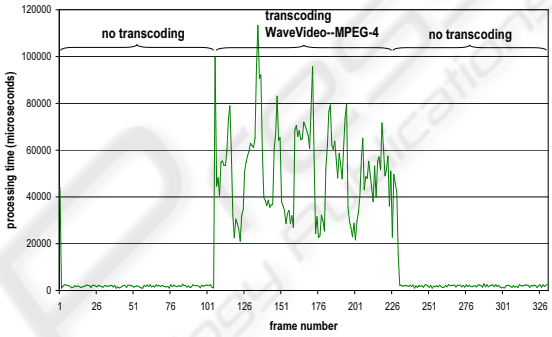

Figure 7: Video transcoding.

3.2.5 Test Sequence 5: Video Transcoding

In a last test sequence, we study the performance of video transcoding. We used a simple spatial domain cascaded video transcoder implementation (see Section 2.2), which decodes the compressed video data completely and re-encodes the uncompressed data using the target format. Figure 7 depicts only the processing time for a single video stream on TS2. The video sequence is encoded with the WaveVideo codec at 20 fps, frame size CIF (Common Intermediate Format = 352 $*$ 288 pixel). On the CAN, the stream is transcoded into MPEG-4 format using the XVID video codec. A maximum of two such streams can be transcoded in parallel on TS2. TS1 does not have enough processing power for transcoding multiple video streams in parallel and in realtime. Even when transcoding one stream, TS1 does not match the framerate of 20 fps. The processing time per video frame is about 50 milliseconds per frame on TS2. When using the smaller QCIF format (Quarter CIF = 176 $*$ 144 pixel), it is possible to transcode up to five streams in realtime on TS2 and up to two streams on TS1.

As described in Section 2.2, performance gains can be achieved by using an optimized video transcoder

instead of the cascaded video transcoder that is used by the current CAN prototype. Sun et al. (Sun et al., 2005) state that highly optimized video transcoders can achieve 70% faster adaptation than a cascaded video transcoder, at the cost of reduced SNR. Nevertheless, even a transcoder that is able to perform the adaptation 70% faster can support only a small number of parallel video transcoding sessions.

4 CONCLUSION

We have seen that many filter techniques are efficient enough to enable real-time adaptation of multiple media streams in parallel on a content adaptation node. We have also seen that content adaptation services benefit from multi-processor or multi-core processor architectures. In general, the gBSD approach requires more processing resources than media filters for specific codecs which do not process gBSD metadata. Even audio transcoding performs slightly better than the implemented gBSD filter. Nevertheless, the MPEG-21 DIA approach has the big advantage that a gBSD adaptation engine can handle any abitrary media format (even not yet developed ones) as long as an appropriate bitstream syntax description of the media stream is available.

For video streams, the temporal performance delta between media filters and transcoders is enormous. In the future, we plan to study the performance of other video transcoders using optimized video transcoding techniques and hardware-supported transcoding. Nevertheless, although noticable performance gains can be achieved by optimized transcoders, it seems very unlikely that in the near future a media gateway will be able to transcode high numbers of high quality video streams in parallel. Video filters will presumably play a more important role in real-time video adaptation than transcoding in the near future.

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

Amir, E., McCanne, S., and Zhang, H. (1995). An application level video gateway. In *Proceedings of ACM Multimedia '95*.

Amon, P. and Pandel, J. (2003). Evaluation of adaptive and reliable videotransmission technologies. In *Proceedings of the 13th Packet Video Workshop*, Nantes, France.

Fankhauser, G., Dasen, M., Weiler, N., Plattner, B., and Stiller, B. (1999). WaveVideo - an integrated approach to adaptive wireless video. *ACM Monet, Special Issue on Adaptive Mobile Networking and Computing*, 4(4).

ISO/IEC (1994). Generic coding of moving pictures and associated audio. International Standard 13818, ISO/IEC JTC1. MPEG-2.

ISO/IEC (2000). Coding of audio-visual objects, part-2 visual. International Standard 14496-2:2001, ISO/IEC JTC1/SC29. MPEG-4.

ISO/IEC (2004). Information Technology - Multimedia Framework (MPEG-21) - Part 7: Digital Item Adaptation. Technical Report 21000-7:2004, ISO/IEC JTC1/SC29/WG11.

Kassler, A., Neubeck, A., and Schulthess, P. (2001). Classification and evaluation of filters for wavelet coded videostreams. *Signal Processing Image Communications*, 16(8):795–807.

Kassler, A. and Schorr, A. (2003). Generic QoS aware media stream transcoding and adaptation. In *Proceedings of the 13th Packet Video Workshop*, Nantes, France.

Keller, R., Choi, S., Decasper, D., Dasen, M., Fankhauser, G., and Plattner, B. (2000). An active router architecture for multicast video distribution. In *Proceedings of INFOCOM*, pages 1137–1146, Tel Aviv, Israel.

McCanne, S., Vetterli, M., and Jacobson, V. (1997). Low Complexity Video Coding for Receiver-Driven Layered Multicast. *IEEE Journal on Selected Areas in Computing (JSAC)*, 14(6):983–1001.

Sun, H., Chen, X., and Chiang, T. (2005). *Digital Video Transcoding for Transmission and Storage*. CRC Press.

Sweazey, P. and Smith, A. J. (1986). A class of compatible cache consistency protocols and their support by the IEEE futurebus. In *Proceedings of the 13th International Symposium on Computer Architecture*.

Vetro, A., Christopoulos, C., and Sun, H. (2003). Video transcoding - architectures and techniques: An overview. *IEEE Signal Processing Magazine*.

Yeadon, N., Garcia, F., Hutchinson, D., and Shepherd, D. (1996a). Filters: QoS support mechanisms for multipeer communications. *IEEE Journal on Selected Areas in Communications*, 14:1245–1262.

Yeadon, N., Garcia, F., Shepherd, D., and Hutchison, D. (1996b). Continuous media filters for heterogeneous internetworking. In *Proceedings of the SPIE Multimedia Computing and Networking*, San Jose, CA.