OPTIMIZED H.264-BASED BITSTREAM SWITCHING FOR WIRELESS VIDEO STREAMING

Thomas Stockhammer, Michael Walter, and Günther Liebl

Nomor Research, Bergen, Germany

ABSTRACT

In this work we show the suitability of H.264/AVC extended profile for wireless video streaming applications. In particular, we exploit the advanced bitstream switching capabilities of H.264/AVC using SP/SI pictures. Optimized encoders for switching pictures are proposed. Finally, a framework for dynamic switching and frame scheduling is proposed and the performance is shown for H.264/AVC video streaming over EGPRS.

1. INTRODUCTION

Adaptivity is one of the most important features for applications in wireless systems to react to the dynamics due to statistical traffic, variable receiving conditions, as well as handovers and random user activity. Due to the applied error control features these variations mainly result in varying bit-rates. However, it is important to understand that the variability cannot be attributed to a single effect, and that the variability also underlies different time scales. The variability of the bit-rate usually scales in different magnitudes, e.g. within a few milliseconds due to short-term fading and interference, within a few hundred seconds due to shadowing effects, within a few seconds due to receiver position as well as within larger scales due to handovers and the overall system load. In case of online encoding and the encoder has sufficient feedback on the expected bitrate on the channel rate control for VBR channels can be applied [1], *ie*, the rate control is adapted to react to changing bit-rates [2].

For pre-encoded sequence other means are necessary. In case of short-scale channel bit-rate variations, playout buffering at the receiver can compensate bitrate fluctuations such that the display timeline is maintained. For example in [3] it has been shown that for UMTS-like channels the bitrate variations due to link layer retransmissions can be well compensated by receiver buffers without adding significant additional delay due to the statistical bitrate fluctuations. In addition, in case of anticipated buffer underrun, techniques such as adaptive media playout [4] allows a streaming media client, without the involvement of the server, to control the rate at which data is consumed by the playout process.

However, in many cases, playout buffering and adaptive media playout might not be sufficient to compensate bitrate variations in wireless channels. In this case the rate adaptation has to be performed by modifying the encoded bit-stream. This adaptation can be carried out at different instances in the network: At the streaming server, in intermediate routers, or the entry gateway to the wireless access network. Usually, one can assume that backbone networks are over-provisioned such that the bottleneck is the wireless link itself. Therefore, it is more likely that closer to the wireless link, there exist more up-to-date channel state information about the expected transmission conditions which would allow making better decisions. On the other hand, the streaming server usually includes much more intelligence to react to variable bit-rates than intermediate routers or gateways. The latter usually only drop packets in case of congestions not taking into account the importance of individual packets usually resulting in error propagation. In this case bitrate adaptivity is equivalent to packet loss resilience - features included in H.264/AVC for this purpose are discussed for example discussed in [5].

However, we assume in the remainder that our rate adaptation entity - referred to as *scheduler* - has sufficient information and intelligence to be able to drop packets with less importance over important packets. A formalized framework under the acronym rate-distortion optimized packet scheduling has been introduced [6] and serves as the basis for several subsequent publications. Obviously, this strategy can be applied on regular syntax by defining more and less important packets. However, anticipating bitrate variations it is common that media streams are pre-encoded with appropriate packet dependencies, such that important and less important packets can be easily differentiated. H.264/AVC provides different approaches to support packets with different importance for bitrate adaptivity though not classical SNR-scalability.

Our proposed streaming system will rely on three different means for bit-rate adaptivity, namely (i) playout buffering (ii) temporal scalability (iii) advanced bitstream switching. We will in the following briefly introduce features for (ii) and (iii) in H.264/AVC. Then, we will discuss suitable encoding and hinting of H.264/AVC streams. Furthermore, we briefly present an optimized decision making on the selection on frames as well as versions and show some experimental results.

2. BITRATE ADAPTIVITY IN H.264

The flexible reference frame concept in combination with generalized B-pictures allows a huge flexibility on frame dependencies which can be exploited for temporal scalability and rate shaping of pre-encoded video. For example, for video including non-reference frames the rate can easily be adapted as dropping of non-reference frames does not result in error propagation. This H.264/AVC operation mode is equivalent to temporal scalability. If frame dropping is not sufficient, one might rely on data partitioning which can be viewed as a very coarse but efficient method for SNR scalability, but this is not considered in the remainder as it results in error propagation.

Also not considered are any means relying on flexible macroblock ordering. Sequences could be encoded such that for example less important background is dropped in favour of a more important foreground scene. However, for many use cases it is still necessary to dynamically adapt the bitrate in the application, usually in larger bitrate scales as well as time scales larger than the initial playout delay. In addition, it has been recognized that the bitrate on wireless links is precious, especially when compared to storage on servers. Finally, most applications provide sufficient buffer feedback as well as channel state information such that the streaming server has at least anticipation on the supported bitrate. Under these common premises in wireless streaming environments bitstream switching provides a simple but powerful mean to support bitrate adaptivity. In this case the streaming server stores the same content encoded with different versions in terms of rate and quality. In addition, each version provides means to randomly switch into it. IDR pictures provide this feature, but they are also costly in terms of compression efficiency.

The switching-predictive (SP)-picture concept in H.264/AVC [7] perfectly fulfills these requirements: In this case the streaming server not only stores different versions of the same content, but also secondary SP-pictures as well as SI-pictures. In case that the bitrate does not change, efficient primary SP-pictures are transmitted. If switching is necessary, one can rely on secondary SP-pictures, or SI pictures as shown in figure 1.

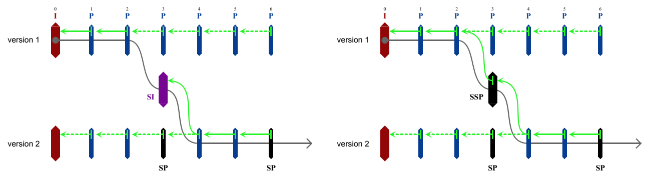

Fig. 1. Bitstream switching with SP and SI pictures in H.264.

3. PREPARING H.264 VIDEO FOR VBR STREAMING

3.1. Encoding

The SP-picture concept allows applying predictive coding even in case of different reference signals by performing the MCP process in the transform domain rather than in the spatial domain and the reference frame is quantized - usually with a finer quantizer than used for the original frame - before it is forwarded to the reference frame buffer. These so-called primary SP-pictures are introduced in the encoded bitstream which are in general slightly less efficient than regular P-pictures but significantly more efficient than regular I-pictures. The major benefit results from the fact that this quantized reference signal can be generated mismatch-free using any other prediction signal. In case that this prediction signal is generated by predictive coding, the picture is referred to as secondary SP (SSP)-picture which are usually significantly less efficient than P-pictures as an exact reconstruction is necessary. To generate this reference signal also without any predictive signal, so-called switching-intra (SI) pictures can be used. SI pictures are only slightly less inefficient than common I pictures and can also be used for adaptive error resilience purposes. For more details on this unique feature within H.264/AVC the interested reader is referred to [7]. An encoder realization of primary SP-pictures is included in the H.264/AVC test model software. In addition, we have implemented an optimized encoder for SSP as well as for SI-pictures. The encoder structure for SSP-pictures is shown in figure 2. As our implementation will serve in a switching scenario, let us assume a simplified switching scenario with only two pre-encoded versions 1 and 2 to be generated. Assume further that both versions have periodically coded primary SP-pictures at identical positions. Thus, at every "SP-position" either the primary is transmitted if no switching happens or the secondary (either SSP

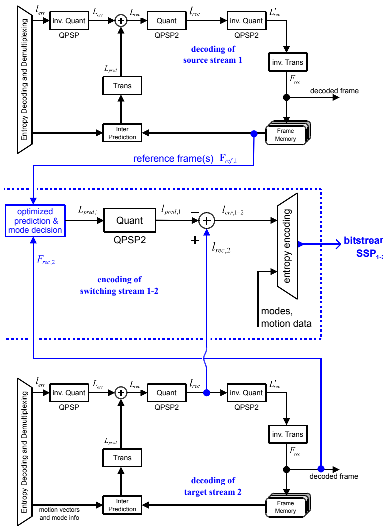
Entropy Decoding and Demultiplexing
inv. Quant
QPSP
Quant
QPSP2
inv. Quant
QPSP2
decoding of source stream 1
inv. Trans
decoded frame
Trans
Inter Prediction
Frame Memory
reference frame(s) $F_{ref,1}$
optimized prediction & mode decision
encoding of switching stream 1-2
entropy encoding
bitstream SSP
modes, motion data
decoding of target stream 2
motion vectors and mode info

Fig. 2. Optimized secondary SP-picture encoder.

or SI) is transmitted in case of switching. Assume that we have encoded two version of the same original video sequence with two different quantization parameters whereby SP-pictures have been used in regular distances at the same positions. According to figure 2 we obtain the SSP-picture for switching from source stream 1 to target stream 2 by extracting and combining information from both runs. The encoding processes for the secondary representations depend on the signal $l_{\text{rec},2}$ that is generated in the encoding and decoding process of the primary target SP-picture. We decided to use the decoding process of target stream 2 for exporting $l_{\text{rec},2}$ as shown in Figure 2. SSP-encoding also requires the prediction signal $L_{\text{pred},1}$. In our implementation, $L_{\text{pred},1}$ is generated using all reference frames, $\mathbf{F}_{\text{ref},1}$, which are available by decoding source stream 1. For SI-pictures the concept applies with the difference that the prediction signal can be computed without any signals exported from stream 1.

The straight-forward approach to simply use the prediction signal, motion vectors and modes from encoding/decoding the primary source stream 1 is not efficient: The partition modes and the motion vectors chosen for encoding the source primary SP-picture do not necessarily fit well for encoding the SSP and result in a sub-optimal prediction signal resulting a large prediction error $l_{\text{err},1\to 2}$. This implies that coding efficiency is low as the residual has to be encoded without any further quantization. Following these considerations, a prediction signal $L_{\text{pred},1}$ is required which minimizes the residual. No restrictions apply on $L_{\text{pred},1}$ so that we can optimize it by using all available reference frames $\mathbf{F}_{\text{ref},1}$. Classical RD optimization [8] as used in the test model is applied. However, the encoded SSP shall be identical to the primary SP-reconstruction of the target stream, the goal of motion estimation and compensation must therefore be to approach the reconstructed primary target frame $F_{\text{rec},2}$, rather than the original frame F_{orig}. With this optimized mode selection we save up to 10% in bits for SSP picture coding compared to the case if we use the prediction signal optimized to F_{orig}. The gains compared to non-optimized approach using the prediction signal $L_{\text{pred},1}$ are in the order of $100-400\%$, the frame sizes often exceed or equal those for SI-pictures. For details on encoding results, further details of the encoder implementation, as well as guidelines on the selection of quantization parameters for primary and secondary representation we refer to [9].

3.2. Media Abstraction - Hinting

Efficient streaming media algorithms require a formalized description of the encoded multimedia data to allow making good decision during the transmission process [6]. Assume that our video frames, a_n, $n = 1, \ldots, N$ are encoded and packetized into *data units* $\mathcal{P}_n$, any advanced packetization modes are not considered. In addition, we assume that for each source unit a_n we generate several versions $\nu = 1, \ldots, V$ which are packetized in distinct data units $\mathcal{P}_{n,\nu}$. The reconstructed version is denoted as $\hat{a}_{n,\nu}$. Furthermore, assume a quality measure $Q(a, \hat{a})$ measuring the rewards/costs when representing a by $\hat{a}$. In addition, each source unit has assigned a decoding time stamp (DTS) T_n representing the time the data unit n must be decoded to be useful. The decoding time is relative to T_1 which is assumed to be 0 without loss of generality. Data units indexes are ordered with DTS T_n. According to [6], video encoding and packetization can be represented as a directed acyclic graph. However, note that this only holds for the data units within one version, the framework of different versions is not addressed in [6]. We restrict ourselves in the following to the scenario where the graph for each version is identical. We write $n' \prec n$ if n' is necessary to decode n.

If we operate in an environment where not all data units are received at the media decoder, concealment has been applied. We use a "freeze-picture" approach and only direct or indirect ancestors are taken as concealing source units. In any case the timely nearest available source unit is used for concealment. The concealment source unit of n is denoted as $c(n)$. If there is no preceding source unit, e.g. I-pictures, we assume that the lost source unit is concealed with a standard representation, e.g. a grey image denoted as $c(n) = 0$. In case of consecutive data unit losses, the concealment is applied recursively. Assume that $c(n) = i$. If data unit i is also lost the algorithm uses source unit j to conceal i, *ie*, $c(i) = j$. To avoid this recursive notion we write $j \vdash n$ meaning source unit n is eventually concealed unit j. Concealment dependencies can also be expressed by a directed graph. Figure 3 shows a frame dependency and a corresponding concealment graph. With these preliminaries, we can express for each data unit

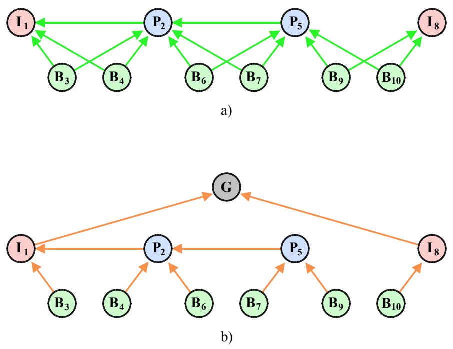

Fig. 3. Frame dependencies and concealment graph.

$\mathcal{P}_n$ the amount by which the quality at the receiver increases if the data unit is correctly decoded, referred to as importance in the following as

$$\begin{aligned} I_{n,v} &\triangleq Q\left(a_n, \hat{a}_{n,v}\right) - Q\left(a_n, \hat{a}_{c(n),v}\right) \\ &\quad + \sum_{\substack{i=n+1 \\ n \vdash i}}^{N} \left[Q\left(a_i, \hat{a}_{n,v}\right) - Q\left(a_i, \hat{a}_{c(n),v}\right)\right]. \quad (1) \end{aligned}$$

When transmitting a stream to a client, a server selects an appropriate version vector $\nu = \{\nu_n\}_{n=1}^{N}$ with ν_n the version chosen for each a_n. Note that there apply restrictions on versions, switching is only possible at I- or SP-picture positions. For details and specific issues on SP-pictures we again refer to [9].

The end–to–end performance of a streaming media system strongly depends on the versions chosen (expressed by the version vector, ν), and, as important the amount and importance of packets being not available at the decoder. To be more specific we define the observed channel behavior at a streaming client for data unit $\mathcal{P}_n$ as $c_n \triangleq \mathbf{1}\{\text{data unit } \mathcal{P}_n \text{ available}\}$ where $\mathbf{1}\{A\}$ defines the indicator function being 1 if A is true and 0 otherwise. In case of a certain observed channel sequence $\boldsymbol{c} = \{c_1, \ldots, c_N\}$ and with the definition in (1) the received quality can be expressed as

$$Q(\boldsymbol{c}, \nu) = Q_0 + \frac{1}{N} \sum_{n=1}^{N} I_{n,\nu_n} c_n \prod_{\substack{m=1 \\ m \prec n}}^{n-1} c_m. \quad (2)$$

with $Q_0 \triangleq \frac{1}{N} \sum_{n=1}^{N} Q\left(a_n, \hat{a}_0\right)$. Although not completely obvious, we skip the proof for conciseness reasons. To summarize, the hinting processes abstracts the encoded source with V versions by generating Q_0, for each data unit $n = 1, \ldots, N$ and each version $v = 1, \ldots, V$ the importance $I_{n,\nu}$, data unit size $R_{n,\nu}$, decoding time stamp T_n, as well as dependencies expressed by the index of the directly preceding data unit(s) of data unit n. Furthermore for each SP-picture in each version v, the size of the SSP, $R_{n,\nu \to \nu'}$, when switching to version v' and the SI-picture size is obtained [9].

4. OPTIMIZED BITSTREAM SWITCHING

Assume now that we transmit data over channel with variable bit-rate whereby the bit-rate variations are not a priori known to the transmitter. Typical examples are wireless bottleneck links employing retransmissions as well as changes in coding and modulation schemes. We have used a model for EGPRS according to [10]. Nevertheless, assume that the server has knowledge about the probability $P(r, t)$ that the channel can successfully transmit r bits within the next t seconds. Note that in general this probability distribution $P(r, t)$ is time-variant.

The video decoder might experience the absence of certain data units $\mathcal{P}_n$ in the decoding process due to (i) losses, e.g. due to channel impairments on the mobile channel or buffer overflows in the network, (ii) delays, i.e. the data unit's decoding time $t_{\text{DTS}}(n) + \Delta$ with Δ the initial playout delay has expired (iii) the server has not even attempted to transmit the data unit. We will assume a persistent mode on link layer which rules out effects due to (i). Therefore, the receiver performance is determined by effects due to (ii) and (iii).

Taking into account the channel behavior $P(r, t)$, the deadlines, the importance, different versions, as well as some state of different data units, the scheduler in the transmitter should decide each time it can transmit a new data unit (i) which data unit to transmit next, possibly out of decoding order, (ii) and in case of an SP or I-picture which version to transmit next. Note that only data units can be transmitted for which all ancestors have been received by the decoder. For example, more "important" data might be transmitted earlier to guarantee their delivery with high probability, whereas other data units with very low importance might not be transmitted at all. This is expressed by the transmission schedule $\boldsymbol{\pi}$ with π_i the data unit transmitted at position i. Also, for each data unit n, a version ν_{π_i} is selected.

During the transmission, we assume that each data unit n has assigned a state γ_n in the transmitter. We define four states, namely $\gamma_n =$ ACK if the data units is known to be received, $\gamma_n =$ NAK if the data unit is known to be not received, $\gamma_n =$ R if the data unit is not yet transmitted, but can be transmitted as all ancestors n' have $\gamma'_n =$ ACK, and finally $\gamma_n =$ P for all data units not yet transmitted and still having some missing ancestors. After having transmitted one data unit n, we assume immediate feedback and update the status of all data units taking into account the new status γ_n as well as the advanced time possibly resulting in expired deadlines of not yet transmitted data units at the transmitter. The selection on the next data unit is based on the expected quality at the receiver, $\mathcal{Q}(\boldsymbol{\pi}, \boldsymbol{\nu})$, taking into account all relevant source and channel information such as rates, deadlines, importance, etc. More specifically, the schedule $\boldsymbol{\pi}_{\text{opt}}$ and the version $\boldsymbol{\nu}_{\text{opt}}$ is selected which maximizes

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\pi}, \boldsymbol{\nu}) &= Q_0 + \sum_{\substack{n=1 \\ \gamma_n = \text{ACK}}}^{N} I_{n,\nu_n} + \sum_{\substack{n=1 \\ \gamma_n = \text{NAK}}}^{N} 0 + \\ & + \sum_{\substack{n=1 \\ \gamma_n = \text{R,P}}}^{N} I_{n,\nu_n} P_n(\boldsymbol{\pi}, \boldsymbol{\nu}) \prod_{\substack{m=1 \\ m \prec n}}^{n-1} P_m(\boldsymbol{\pi}, \boldsymbol{\nu}). \quad (3)\end{aligned}$$

whereby $P_n(\boldsymbol{\pi}, \boldsymbol{\nu})$ expresses the probability that data unit n will be received in time for this selection of $\boldsymbol{\pi}$ and $\boldsymbol{\nu}$. Note that for data units with $\gamma_n =$ ACK this probability is 1, for $\gamma_n =$ NAK it is 0. In the case of $\gamma_n =$ R,P the probability depends on the sum rate of the scheduled data units, the delivery deadline $T_n + \Delta$ of data unit n, the actual time, and the channel statistics $P(r, t)$.

Finding the optimum schedule and optimal version is a complex task, as brute-force search is basically necessary. However, we have developed methods, which significantly simplify the search by only taking into account very few data units and versions in the scheduling process and fixing the transmission position for later data units. The algorithm basically selects the next data unit to be transmitted, possibly not transmitting some non-important data units. In addition, it also takes into account that versions can be switched. We define the number of data units considered for scheduling as *look ahead units*, $N_{\mathcal{P}}$, and the number of switching points considered in the decision as *look ahead switching points*, N_S. Obviously, the more data we take into consideration, the more complex the algorithm gets, but also the performance gets better. For details such as complexity reduction, integration of SP and SI-pictures, etc. we refer again to [9] and references therein.

5. EXPERIMENTAL RESULTS

We have encoded $V = 4$ versions of a QCIF sequence of length $N = 2698$ with alternating speakers and sport scenes using H.264/AVC test model software JM8.2 applying a single QP for each version, namely QP $= 28, 32, 36, 40$ at frame rate 30 fps without using any rate control algorithm. A Group–of–Picture (GOP) structure with IBBPBBP...SP has been applied with SP-picture distance of 1 s. The SP–pictures have "Instantaneous Decoder Refresh (IDR)" property in a sense that referencing over SP–pictures is not permitted. In addition, SSP-pictures and SI-pictures are provided, the initial playout delay is $\Delta = 1.5$ s.

Selected results for different streaming technologies are shown in figure 4. For an EGPRS channel combining 3 transmission slots the bit-rate varies in the range of about 65 to 120 kbit/s. In case that we stream a fixed bit-rate stream we still apply the optimization in 3, but only for one version resulting optimized temporal scalability is applied. We show the average PSNR for different look-ahead switching points. Obviously if we look further into the future, we obtain better results as we can take into account the statistics of future data units in the decision. The streaming performance for two selected streams with fixed bit-rate yield that the low bit-rate case provides constant quality as basically all data units are received. For the higher bit-rate stream, we observe improvements, but it is obvious that the overall quality is low as very often frames are not even transmitted. When applying bit-stream switching with I-pictures we can gain about 0.5 dB for $N_s \geq 3$, with the inclusion of SP-pictures another 0.5 dB are achievable. The good performance of bitstream switching is shown, but we also observe that optimization in the streaming scheduler are essential to exploit the potential of bitstream switching, in particular the SP-picture concept in H.264/AVC. The system still provides a significant amount of optimization potential which is subject of ongoing work. For more details we again refer to [9].

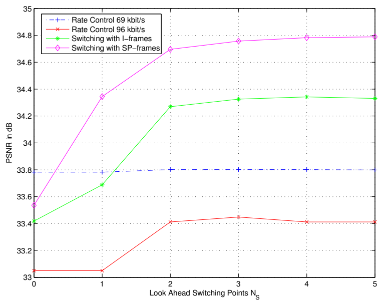

Fig. 4. Bitstream switching versus rate control on EGPRS.

6. REFERENCES

[1] A. Ortega and K. Ramchandran, "Rate–distortion methods in image and video compression," *IEEE Signal Processing Mag.*, vol. 15, no. 6, pp. 23–50, Nov. 1998.

[2] C.Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 5, pp. 756–773, May 1999.

[3] T. Stockhammer, H. Jenkac, and G. Kuhn, "Streaming video over variable bit-rate wireless channels," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 268–277, Apr. 2004.

[4] M. Kalman, E.G. Steinbach, and B. Girod, "Adaptive media playout for low–delay video streaming over error–prone channels," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 841–851, June 2004.

[5] S. Wenger, "H.264/AVC over IP," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 645–656, July 2003.

[6] P. A. Chou and Z. Miao, "Rate–distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, Feb. 2001, Submitted. http://research.microsoft.com/ pachou.

[7] M. Karczewicz and R. Kurceren, "The sp and si frames design for h.264/avc," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 13, no. 7, July 2003.

[8] T. Wiegand, M. Lightstone, D. Mukherjee, T.G. Campbell, and S.K. Mitra, "Rate–distortion optimized mode selection for very low bit rate video coding and the emerging h.263 standard," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 6, pp. 182–190, Apr. 1996.

[9] M. Walter, "Advanced bitstream switching for wireless video streaming," Diploma thesis, Munich University of Technology (TUM), Dec. 2004, available at http://www.nomor.de/publications/Documents/DA_MichaelWalter.pdf.

[10] Jian Cai, Li Fung Chang, Kapil Chawla, and Xiaoxin Qiu, "Providing differentiated services in EGPRS through packet scheduling," in *IEEE Globecom*, Nov. 2000.