# Bit-error Resilient Packetization for Streaming H.264/AVC Video

Jari Korhonen
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne

Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne

## ABSTRACT
Bit errors in the radio channel can severely decrease the efficiency of the link utilization in wireless multimedia communications. One possibility to avoid consuming bandwidth with numerous packet retransmissions and thus improve the performance of a wireless video streaming system is to pass the packets to the application layer, even if they are corrupted by bit errors. In this case, the application layer should be capable of using partially damaged video data. Unfortunately, the state-of-the-art video coding standards do not fully support bit error resiliency. In this paper, we propose bit error resilient packetization strategies for streaming H.264/AVC video, based on the use of small slices and protection of the most vulnerable bits only using the UDP-Lite protocol. The simulation results show that the resilience against bit errors can be increased with only slightly decreased compression efficiency, which results in improved streaming performance.

## Categories and Subject Descriptors
H.4.3 [**Information systems applications**]: Communications applications – *Computer conferencing, teleconferencing, and videoconferencing.*

## General Terms
Design, Measurement, Performance.

## Keywords
Video streaming, Bit error resilience, H.264/AVC, UDP-Lite.

## 1. INTRODUCTION
Due to the error prone nature of a typical wireless communication channel, error recovery represents an important problem in multimedia delivery in wireless networks. Basically, there are two approaches to recover from transmission errors. First, the receiver application can try to reconstruct the missing or damaged parts of the video stream without any support from the sender application, for example by interpolation from the correctly received neighboring parts of the stream (Error Concealment, EC). The second strategy relies on channel coding or joint source and channel coding, and attempts to correct the missing parts of data by using the redundant information that have been added to the video stream (e.g., Forward Error Correction, FEC) or requesting retransmissions for the lost data sections (Automatic Repeat reQuest, ARQ). FEC and ARQ schemes cause transmission overhead and latency. Efficient strategies therefore result from an effective compromise between redundancy for error resilience and overall bandwidth requirements.

In wireless multimedia communications, the last hop in the radio access network is typically the bottleneck of transmission. Because radio communications suffer from bit errors especially at high transmission rates, robust modulation schemes and link layer retransmissions can increase the link occupancy level significantly and thus decrease the effective data rates for the mobile users. One possible approach to reduce the retransmission overhead is to pass the damaged packets up to the application layer as such, instead of retransmitting them [3,4]. Of course, in this case the application layer must be capable of detecting errors and utilizing the partially damaged data. For this purpose, bit error resilient coding techniques would be beneficial.

Error concealment mechanisms are typically best applicable if the damaged regions are small. For example, a large number of small erroneous sections distributed smoothly over the video stream in both spatial and temporal dimensions typically result in better perceived quality than smaller number of large errors. This is the rationale behind certain techniques facilitating error concealment, such as slice interleaving and Flexible Macroblock Ordering (FMO) in H.264/AVC [1,2]. Due to the highly hierarchical data structure present in the state-of-the-art video codecs, a small error may still lead to severe error propagation, even in case these techniques are utilized.

In this paper, we propose to increase the robustness to bit errors, when corrupted data packets are passed to the application layer in order to limit the number of retransmissions. We study the bit error resilience and robust packetization strategies for streaming video. We propose new schemes for packetizing decodable elements of a video stream into RTP packets, such that the most important information receives prioritized treatment, and the distortion in case of bit error stays limited. We verify the performance of the proposed approaches in a simulation study using the recent and popular H.264/AVC codec.

The rest of the paper is organized as follows. In Section 2, we give a brief overview to the error resilience in multimedia communications. In Section 3, the proposed packetization scheme


*MV'07*, September 28, 2007, Augsburg, Bavaria, Germany.

is explained and its performance is evaluated. Finally, the conclusions are given in Section 4.

## 2. ERROR RESILIENCE

Several mechanisms and tools to improve the error resilience in advanced multimedia coding technologies have been proposed and even adopted in standards. In general, these mechanisms target either data losses or bit errors. In the first case, it is assumed that some data elements may be lost during the transmission, but the received data is definitely free of errors. In the latter case, some bits in the received data can be erroneous and therefore inconsistent with the bitstream syntax.

### 2.1 Packet Loss Resilience

Typically, advanced multimedia coding standards and packet payload formats allow detection of lost data elements by using sequence numbers. When a packet loss is detected, appropriate error concealment can be applied to replace the missing data. When packet losses are considered, each transport packet contains ideally an integral number of individually decodable data elements, because fragmented elements may become entirely useless if just one of the fragments is lost.

In H.264/AVC, the basic element for decoding is called Network Abstraction Layer (NAL) unit. One NAL unit (NALU) may contain decoding parameters or a slice of picture data for either predicted or intra frames. Each slice comprises one or more macroblocks of 16x16 pixels [2,5]. Basically, the slice size can be selected rather freely from one macroblock even up to the all macroblocks in the frame. The damage caused by each packet loss can be minimized if small NALUs are used, and there is only one NALU in each transport packet. However, the use of small slices decrease coding efficiency, due to overhead from NALU headers and transport protocol headers. Even if several small NALUs are packed in one transport packet, some control information is added in the payload for each NALU. Therefore, NALU size is a trade-off between error resilience, bitrate of the transport stream and coding efficiency.

H.264/AVC standard includes a data partitioning tool that can be used to allocate the data components of one slice in three different priority NALU types (A, B, and C) [2,5]. When this technique is utilized, it is straightforward to apply unequal error protection for different partitions. For example, packets containing class A NALUs can be protected by FEC or retransmissions, whereas packets with class B and C NALUs can get weaker protection, or even be left unprotected. B and C partitions are useless without the corresponding A partition.

### 2.2 Bit Error Resilience

As variable length coding is usually applied in advanced multimedia compression technologies, even a single bit error can mutate the length of the codeword seen by the decoder, causing the decoding of the following codeword start in a wrong position. This may lead to severe error propagation within the decodable unit of data. A similar effect may result when a value of a critical flag is flipped. If the decoder is not designed to detect illegal codewords and prevent reading data beyond the borders of the input buffer, decoder may even crash. Even if these error checks are performed, misinterpreted data may severely damage to the resulting quality of the decoded stream if errors are not detected. This is why data losses may be preferred to bit errors.

Several different mechanisms have been proposed to improve the resiliency against bit errors in multimedia coding. The most vulnerable parts of data are usually protected by FEC. Error propagation in variable length coded data can be limited by adding certain delimiters or using more robust codeword sets, such as Reversible Variable Length Codes (RVLC). In RVLC, each codeword can be read from both directions, backward or forward. If an illegal codeword is detected, decoding process can continue from the end of the sequence [6]. All these methods add overhead or decrease the compression efficiency, and are not even used in the most recent versions of video coding standards.

**a) Bit errors in an I-frame**

**b) Bit errors in a P-frame**

**Figure 1. Bit errors in different types of video frames.**

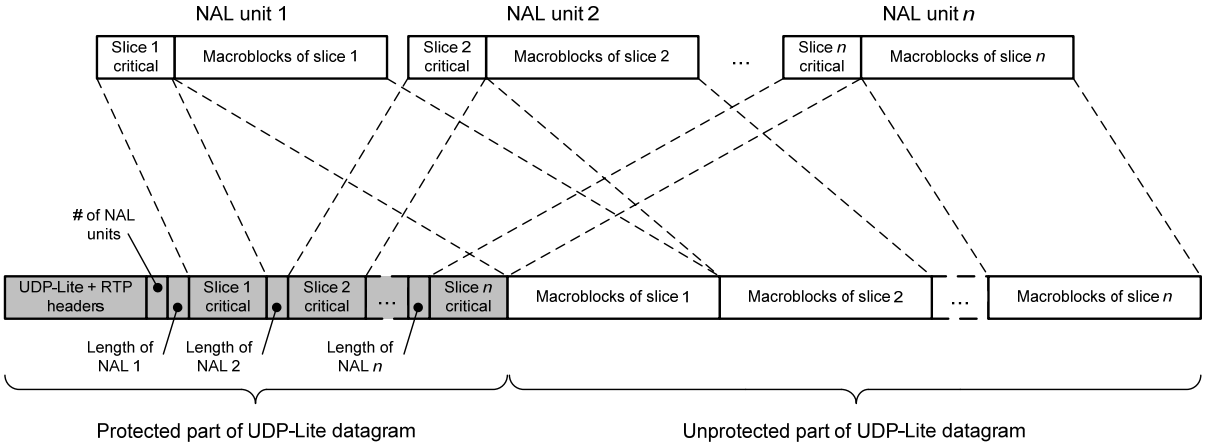
One interesting method adopted in the error resilience tools of MPEG-4 advanced audio coding (AAC) is Huffman code reordering [7]. In this method, priority codewords are allocated in

predefined positions. The remaining codewords are written so that they fill the gaps left between the priority codewords. In this way the error propagation between priority codewords can be avoided and only the lower priority codewords would be affected. Basically, the data partitioning tool could be used to make the distinction between sensitive and non-sensitive bits in H.264/AVC. However, since the data partitioning tool has not been designed for bit error resilience, its efficiency stays limited in the case of bit errors. For example, Masala et. al. have proposed a scheme with link layer partial checksum to protect class A partitions, but they have decided to discard erroneous B and C partitions directly, even if the related A partition is correctly received [8].

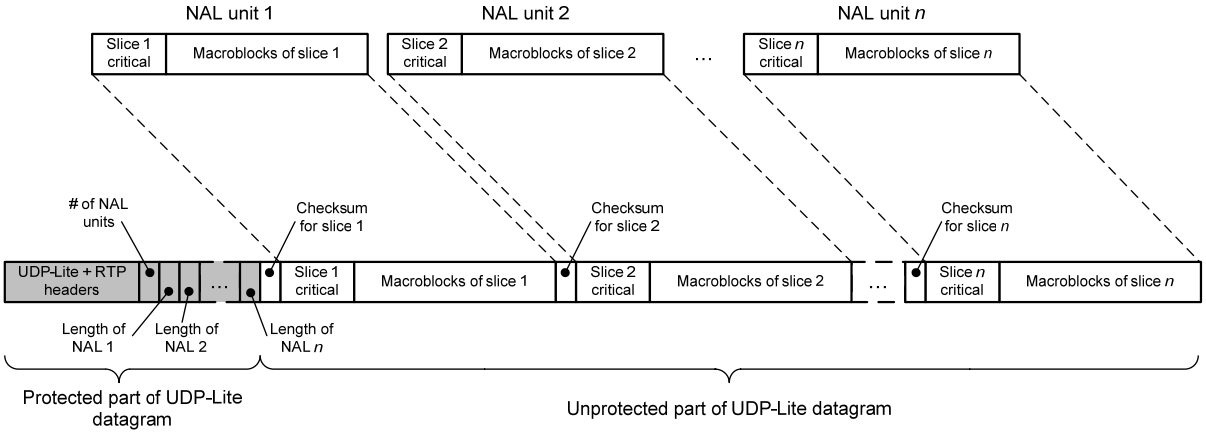
Unfortunately, H.264/AVC does not support bit error resilience at syntax level. However, if only the most sensitive data is protected, a reasonable degree of bit error resilience can still be achieved by detecting syntax violations and illegal values of decoded data elements. When an error is detected, damaged macroblocks can be repaired by using traditional error concealment schemes. Non-detected errors may cause misinterpretations of data, such as wrongly decoded motion vectors in predicted frames or distorted chroma components in intra frames. Figure 1 shows examples of effects caused by bit errors in H.264/AVC coded I- and P-frames. Typically, errors in I-frames are perceptually different than in P-frames, because macroblocks with erroneous chroma components appear as clearly visible squares (see Figure 1a) and erroneous motion vectors appear as misplaced blocks (see Figure 1b).

# 3. RESILIENT TRANSPORT OF VIDEO

Usually, in wireless packet-switched networks bit errors are detected at the link layer already, and erroneous packets are discarded and requested to be retransmitted. However, several studies show that a considerable amount of wireless link bandwidth can be saved if erroneous packets are passed up to the application instead of retransmitting [9, 10, 11]. Of course, in this




**a) Bit error resilient packetization scheme for UDP Lite**




**b) Bit error resilient packetization scheme including checksums for individual slices**

**Figure 2. Bit error resilient packetization strategies for short NALUs in UDP-Lite packets.**

case the application must be able to deal with erroneous data. This is the problem we address with the bit error resilient packetization strategies proposed below.

In a typical case, packet header and the most critical parts of multimedia data must be protected against bit errors even in case errors are allowed in some parts of the data. For this purpose, UDP-Lite protocol [4] has been proposed. Assuming that the erroneous packets would reach the transport layer, UDP-Lite uses a partial checksum to cover only the packet header and the critical data of the payload located in the beginning of the packet. If a bit error is detected in the protected part, the whole packet is discarded. Otherwise, it is passed further up to the application.

When a packet contains several individually decodable data units, it is necessary to protect the vulnerable data of each of these units either by using application layer checksums or reallocating the vulnerable data in the beginning of the RTP payload and covering it by the UDP-Lite checksum. An example of such packetization is the payload format for AMR-WB speech codec, where the table of contents with the critical information (frame types) is allocated in the beginning of the payload, separately from the actual speech data [12].

Figure 2a illustrates a bit error resilient packetization strategy for H.264/AVC video. The NALUs (slices) are split in two parts, the first part containing the most relevant bytes, including the slice header. Then, the protected part of each slice is allocated in the beginning of the packet payload, preceded by the length information of the slice. This is how the most vulnerable data of each slice can be protected by the UDP-Lite checksum. The macroblock data resides in the unprotected part.

If it is not possible to implement bit error resilience features in the decoder, it would be possible also to use checksums covering the data for each slice. Together with the slice length information in the protected area, erroneous slices could be detected before decoding and skipped. In this case, only the length information needs to be allocated in the protected area. The setback of this approach is that the possibly unharmed macroblocks in the beginning of each slice are lost. This kind of strategy is illustrated in Figure 2b. It is worth noting that in this case there is no need to protect the NALU headers separately. This is why shorter protected portion of UDP-Lite packets can be used and further reduction in transport layer packet loss rate can be achieved.

# 4. ANALYSIS

## 4.1 Simulation Setup

The performance of the proposed packetization strategy has been evaluated by simulating a UDP-Lite transport system and a wireless channel with a two-state Gilbert-Elliot model. The packetization strategies used were similar to those shown in Figure 2. For the sake of simplicity, we have not used data partitioning in our experiments, but the protected section is heuristically defined as six bytes from the beginning of a NALU. In the simulations, it is assumed that the first reference frame and the protected parts of the UDP-Lite payloads are always received correctly. In a real system, this can be achieved by using retransmissions for the entirely lost packets. The precise implementation details and performance analysis of the network protocols are topics for future research.

In the simulations, JM reference codec version 11.0 [13] was used. Because syntax level bit error resilience is not supported in H.264/AVC, we have modified the reference codec by adding simple error detection functionality. If the decoder observes an illegal codeword or reads beyond the end of the input buffer, the remaining macroblocks in the slice are marked as corrupted and recovered by the standard error concealment (motion vector copy). Two CIF test sequences of 90 frames were used, relatively static 'Foreman' and more demanding 'Soccer', both encoded with target bit rate of 512 kbit/s, (30 frames/s, every ninth frame is an I-frame, B-frames are not used).

Errors have been applied to the unprotected parts of the UDP-Lite payloads generated from the encoded bitstream. Then, the erroneous bitstreams have been decoded and analyzed by measuring the PSNR compared to the original video. There are three different versions of the encoded bitstream, with different maximum NALU sizes (150, 450 and 1350 bytes), packed in UDP-Lite datagrams with maximum payload size of 1400 bytes. Depending on the case, there are from one to ten NALUs per packet, and it is worth noting that if traditional UDP was used instead of UDP-Lite, loss of one packet would lead to loss of all these NALUs. To verify the generality of the results, some experiments were made also with lower bitrate (256 kbit/s).

Tables 1 and 2 summarize the characteristics of the encoded bitstreams for 'Foreman' and 'Soccer' sequences, respectively. The best PSNR value is achieved when the NALU size is largest, reflecting the reduction in compression efficiency when small NALUs are used. However, the difference in PSNR between NALUs of 150 bytes and 1350 bytes is small, around 0.5 dB, in both cases. Because NALUs are not fragmented, the average packet size is larger for the small NALUs providing finer granularity. This reduces slightly the packet header overhead with small NALUs and compensates partially the loss of compression efficiency.

**Table 1. Characteristics of the encoded 'Foreman' sequences**

| | FM150 | FM450 | FM1350 |
|---|---|---|---|
| PSNR | 38.78 | 39.12 | 39.32 |
| Avg. NAL size | 141 B | 399 B | 1004 B |
| Avg. packet size | 1397 B | 1284 B | 1095 B |
| NALs per packet | 9.91 | 3.22 | 1.09 |

**Table 2. Characteristics of the encoded 'Soccer' sequences**

| | SO150 | SO450 | SO1350 |
|---|---|---|---|
| PSNR | 36.33 | 36.67 | 36.84 |
| Avg. NAL size | 147 B | 404 B | 967 B |
| Avg. packet size | 1394 B | 1275 B | 1087 B |
| NALs per packet | 9.96 | 3.15 | 1.12 |

Bit errors were generated using a two state model with three different parameter sets as given in Table 3. The model has two states, the good state and the bad state. In the good state, all bits are transmitted correctly. In the bad state, the transmitted bit is corrupted at probability $P_{b\_err}$. Transition probabilities between the two states are marked with $P_{g\_b}$ from the good state to the bad state, and $P_{b\_g}$. vice versa. In the first test set, bit errors are spread purely randomly. In the second set, there are short error bursts at

relatively high density. In the third set, error bursts are longer, but they appear at longer intervals. The resulting average proportion of erroneous bits in all cases is approximately $10^{-3}$, even though the distribution of bit errors is different.

**Table 3. Parameters used for the two-state model**

| | $P_{g\_b}$ | $P_{b\_g}$ | $P_{b\_corr}$ |
|---|---|---|---|
| 1. Random | $10^{-3}$ | 1 | 1 |
| 2. Short bursts | $10^{-4}$ | 0.05 | 0.5 |
| 3. Long bursts | $2 \cdot 10^{-5}$ | 0.015 | 0.5 |

## 4.2 Simulation Results

Average PSNR values for each test case are shown in Table 4 for 'Foreman' sequence and Table 5 for 'Soccer' sequence, respectively. In addition, Table 6 shows the results for the low bitrate version of the 'Foreman' sequence. For each bitstream with different maximum NALU sizes, two error management strategies were tested: erroneous NALUs are dropped and concealed with traditional means (loss), or NALUs are decoded with the modified decoder that has bit error robustness features (err). In practice, the loss scenarios with large NALUs (1350 bytes) are roughly comparable to a baseline scenario without smart packetization and only one NALU per packet. In some cases, when there are several errors in the long NALUs, the number of entirely discarded frames becomes so high that reliable PSNR value could not be derived. These cases are marked with 'X'.

The results show that bit error resilient decoding outperforms discarding the damaged NALUs in most cases; the only exception is observed when the shortest NALUs are used over a channel with highly bursty bit error pattern. Even is this case, the difference is small. Short NALUs give better results than long NALUs, and long bit error bursts at low density have smaller impact than random bit errors or short bursts at high density, indicating that one bit error within a slice may be almost as harmful as a burst of bit errors.

The results are well in line with intuitive analysis. Larger NALUs are more likely to be hit by bit errors than small NALUs; this is why the proportion of erroneous slices can be reduced by using small NALUs. If there are bit errors in a slice comprising several macroblocks, all the macroblocks prior to the first appearance of an error can be correctly decoded. This is why it is worthwhile to attempt decoding the damaged frames instead of discarding them immediately. However, decoder is not always able to detect erroneous codewords and, in some cases, undetected errors may result in severe distortion in decoded macroblocks. This is why checksum-based approach performs better for some frames even though the performance in general is better with the decoding approach.

Figure 3 shows an example of the PSNR values over Soccer sequence with NALUs of 150 bytes (upper thick curves) and NALUs of 450 bytes (lower thick curves). The solid thick line shows the PSNR values when frames containing short bit error bursts are decoded, and the dotted line shows the PSNR values when the affected frames are dropped instead of decoding. The solid thin curves show the PSNR values of the encoded streams when no error occurs. Under error free conditions, the two curves are almost identical, showing that the compression efficiency is not radically decreased even if small slices are used. When there are errors present, the PSNR curve contains clearly noticeable "spikes" at the positions of I-frames. This indicates that high frequency of I-frames is desirable, when there are errors in the bitstream.

**Table 4. PSNR values for 'Foreman' sequence (512 kbit/s)**

| | No bit errors | Random errors | Short bursts | Long bursts |
|---|---|---|---|---|
| FM150-loss | 38.78 | 22.83 | 31.72 | **35.54** |
| FM150-err | 38.78 | **25.54** | **32.07** | 35.25 |
| FM450-loss | 39.11 | X | 26.45 | 29.99 |
| FM450-err | 39.11 | **21.82** | **29.00** | **32.33** |
| FM1350-loss | 39.32 | X | X | 27.34 |
| FM1350-err | 39.32 | **18.74** | **25.66** | **29.03** |

**Table 5. PSNR values for 'Soccer' sequence (512 kbit/s)**

| | No bit errors | Random errors | Short bursts | Long bursts |
|---|---|---|---|---|
| SO150-loss | 36.34 | 22.94 | 30.51 | **33.68** |
| SO150-err | 36.34 | **24.50** | **30.77** | 32.97 |
| SO450-loss | 36.67 | X | 25.46 | 29.90 |
| SO450-err | 36.67 | **21.88** | **26.75** | **31.62** |
| SO1350-loss | 36.84 | X | X | 24.44 |
| SO1350-err | 36.84 | **20.35** | **23.57** | **26.28** |

**Table 6. PSNR values for 'Foreman' sequence (256 kbit/s)**

| | No bit errors | Random errors | Short bursts | Long bursts |
|---|---|---|---|---|
| FM150-loss | 35.99 | 21.81 | 30.37 | **33.50** |
| FM150-err | 35.99 | **25.02** | **31.23** | 33.48 |
| FM450-loss | 36.53 | X | X | 30.42 |
| FM450-err | 36.53 | **20.99** | **27.84** | **30.99** |

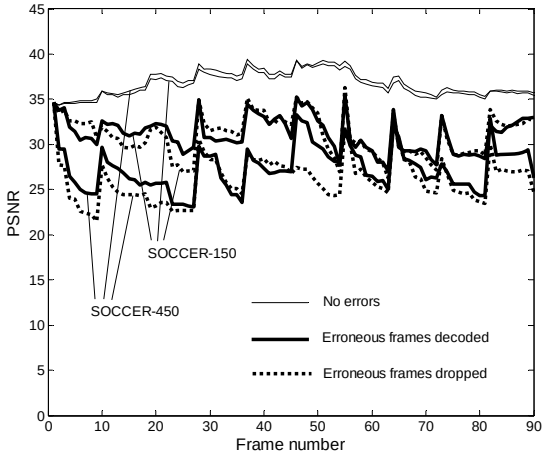



**Figure 3. PSNR values in erroneous 'Soccer' sequences.**

Figure 4 illustrates the difference between these two error management strategies in practice: in Figure 4a, a frame of the Foreman sequence (slices of 450 bytes) containing bit errors is decoded as such. In Figure 4b, the affected slices are discarded and reconstructed with the standard error concealment scheme in the decoder. As we can see, in the first case the damaged areas are smaller, as only part of the macroblocks within each slice are impacted, but the distortion in these areas is more severe. In contrast, in the latter case, the distorted areas comprise entire slices. In our experiments, FMO is not used. FMO would be useful to alleviate the clusterization of damaged macroblocks; however, the total number of damaged or missing macroblocks would not change.


**a) Erroneous slices decoded**


**b) Erroneous slices discarded and concealed**

**Figure 4. Different error management strategies compared.**

## 5. CONCLUSIONS

In this paper, advanced bit error resilient packetization strategies have been proposed for transporting H.264/AVC video over a wireless channel utilizing UDP-Lite. It is shown that by packing several small NALUs in each RTP packet and using a bit error resilient decoder or individual checksums to detect errors in each NALU, reasonable robustness against bit errors can be achieved. According to several other studies, capacity of a wireless link can be used much more efficiently if erroneous packets are conveyed to the application instead of discarding and retransmitting.

However, practically all wireless standards perform error detection already at the link layer. The results suggest that significant benefits could be gained by allowing delivery of partially corrupted packets. This should be taken into account into the development of wireless streaming solutions, in particular in promising cross-layer architectures.

## REFERENCES


[1] Varsa, V., and Karczewicz, M., "Slice Interleaving in Compressed Video Packetization", Packet Video Workshop, Forte Village, Italy (2002).

[2] Wenger, S., "H.264/AVC over IP," *IEEE Trans. Circuits and Systems for Video Techno*logy, vol. 13, no. 7, pp.645-656 (2003).

[3] Weltzl, M., "Passing Corrupt Data Across Network Layers: An Overview of Recent Development and Issues", *EURASIP Journal on Applied Signal Processing*, vol., no. 2, pp. 242-247 (2005).

[4] Larzon, L., Degermark, M., Pink, S., Jonsson, L., and Fairhurst, G., "The Leightweight User Datagram Protocol (UDP-Lite)", IETF RFC 3828 (2004).

[5] ITU-T, "Advanced Video Coding for Generic Audiovisual Services," ITU-T Recommendation H.264 (2005).

[6] Takishima, Y., Wada, M., and Hurakami, H., "Reversible Variable Length Codes," *IEEE Trans. Comm.*, vol. 43, no. 2/3/4, pp 158-162 (1995).

[7] Sperschneider, R., Homm, D., and Chambat, L., "Error Resilient Source Coding with Variable-Length Codes and its Application to MPEG Advanced Audio Coding," 109th Audio Engineering Society Convention, Los Angeles, California, USA (2000).

[8] Masala, E., Bottero, M., and De Martin, J.C., "Link-Level Partial Checksum for Real-Time Video Transmission over 802.11 Wireless Networks," Packet Video Workshop, Irvine, California, USA (2004).

[9] Singh, A., Konrad, A., and Joseph, A., "Performance Evaluation of UDP Lite for Cellular Video," NOSSDAV, Port Jefferson, New York, USA (2001).

[10] Khayam, S., Karande, S., Radha, H., and Loguinov, D., "Performance Analysis and Modeling of Bit Errors and Losses over 802.11b LANs for High Bit-Rate Real-Time Multimedia," *Signal Processing: Image Communication*, vol. 18, no. 7, pp. 575-595 (2003).

[11] Vehkaperä J., Peltola, J., Huusko, J., Myllyniemi, M., and Majanen, M., "Evaluation of Achieved Video Quality in Wireless Multimedia Transmission using UDP-Lite," IASTED IMSA, Honolulu, Hawaii, USA (2006).

[12] Sjöberg, J., Westerlund, M., Lakaniemi, A., and Xie, Q., "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs," IETF RFC4867 (2007).

[13] H.264/AVC Reference Software Archive. Available online: http://iphome.hhi.de/suehring/tml/download /old_jm/.