Error control for Video Streaming with Small Data Units

Jari Korhonen[1]
Norwegian Univ. of Science and Technology (NTNU)
Centre of Quantifiable Quality of Service (Q2S)
NO-7491 Trondheim, Norway
jari.korhonen@q2s.ntnu.no

Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory (LTS4)
CH-1015 Lausanne, Switzerland
pascal.frossard@epfl.ch

ABSTRACT
In multimedia streaming, small errors are typically easier to mask with common error concealment strategies, but small packet size increases the overhead caused by network header information. To reduce the header overhead, large packets are typically favored. In multimedia streaming applications, every packet comprises ideally one individually decodable data unit only. Unfortunately, large packets penalize the error concealment performance at the decoder, which may lead to large and fluctuating distortion. In this paper, we propose an error control mechanism based on efficient packetization of small independent decoding units. Instead of using erasure correction codes to protect packets as such, it gathers several small source data units in each transport packet together with redundancy data units, and the distribution of the units is chosen in order to minimize the distortion at the decoder. The proposed technique has been evaluated by simulating an H.264/AVC video streaming system and comparing the performance against conventional erasure protection scheme involving large data units. The results show that in the presence of packet losses the proposed mechanism provides smoother perceived video quality degradation performance than the conventional packetization and generic forward error correction mechanisms.

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

General Terms
Algorithms, Performance, Experimentation.

Keywords
Video Coding, Multimedia Streaming, Forward Error Correction.

Mobimedia 2008 July 7-9, 2008, Oulu, Finland.

DOI 10.4108/ICST.MOBIMEDIA2008. 3910

1. INTRODUCTION
Basically, there are two basic approaches to recover from packet losses in video communications over packet-switched networks. First, the receiver application can try to reconstruct the missing or damaged parts of the video stream, possibly without any support from the sender application, interpolating them from the correctly received neighboring parts of the stream. This strategy is often referred as *error concealment* (EC), as it intends to conceal the perceptual impact of errors instead of actually correcting them. Another strategy is to regenerate the missing parts of data by using added redundant information in the transport stream (*forward error correction*, FEC) or requesting retransmissions for the lost data (*automatic repeat request*, ARQ).

Due to timing constraints and inadequate support for retransmissions in multi-user applications, ARQ is often not a viable option for error recovery in multimedia communications. FEC is therefore usually preferred in most of the low delay streaming applications that can afford some packet loss. Generally speaking, FEC can be used to decrease the observed residual loss rate of source data blocks or symbols, but it cannot guarantee fully reliable delivery of data. This is why FEC and EC are typically used to complement each other. Error concealment mechanisms work generally best if the lost regions are small, since adjacent video information can be used for masking the damaged areas. In particular, a large number of small lost sections distributed smoothly over the video frames often result in better perceived quality than smaller number of large losses. This is the rationale behind several interleaving mechanisms and techniques facilitating error concealment, such as *flexible macroblock ordering* (FMO) standardized in H.264/AVC [1, 2].

Ideally, each transport packet contains a single individually decodable data unit. Fragmented units may become entirely useless if just one of the fragments is lost, and on the other hand, it would be desirable to allocate as little data in each packet as possible in order to minimize the impact of a single packet loss [3]. Unfortunately, small packet payload leads to large header overhead and inefficient use of the transport channel capacity [1]. This is why packetization techniques always present a compromise between bandwidth efficiency and resilience against packet losses. In this paper, we propose a scheme that aims to alleviate the problem with large losses without increasing the packet header overhead. Several small and independently

[1]This work was partially carried out at during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

decodable data units are gathered in a network packet, in addition to FEC redundancy data units. An efficient technique for distribution of the data units in the transmission packets is proposed in order to minimize the distortion caused by the loss of any combination of transport packets.

The rest of this paper is organized as follows: Section 2 summarizes the background and the relevant related work. Section 3 describes the proposed packetization and FEC mechanism in details. The scheme is evaluated and compared against traditional packetization in Section 4. Finally, the concluding remarks are given in Section 5.

2. BACKGROUND AND RELATED WORK

2.1 Video Coding and Error Resilience

In H.264/AVC, the basic element for decoding is called *network abstraction layer unit* (NALU) [1]. One 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. Basically, the slice size can be selected rather freely from one macroblock even up to the all macroblocks in the frame.

Typically, lost NALUs are detected by using sequence numbers. When a loss is observed, appropriate error concealment can be applied to replace the missing data. The simplest method is to repeat the latest correctly received slice or frame, but better results can be achieved with more advanced algorithms. Even though error concealment is not formally included in H.264/AVC, non-normative error concealment features have been defined for the standard. For self-contained intra frames (I-frames), weighted pixel value averaging can be used to interpolate each pixel in a missing macroblock using the pixel values from the correctly decoded neighboring macroblocks. For predicted frames (P- and B-frames), it is possible to predict the motion vectors of lost macroblocks by analyzing the motion activity in the correctly received slices [4].

To facilitate error concealment, the H.264/AVC standard includes several tools, such as *flexible macroblock ordering* (FMO) [1]. With FMO, the macroblocks within a frame can be interleaved or scrambled so that adjacent macroblocks will be allocated in different NALUs. In this way, the probability of losing adjacent macroblocks can be reduced. There are several possible interleaving and shuffling patterns that can be used for FMO. In practical experiments, FMO with a simple checkerboard pattern (dispersed mode) has been reported to improve the average performance significantly when packet losses occur [5].

Data partitioning tools have also been proposed in H.264/AVC, where it is possible to generate partitions with different perceptual importance (A, B and C partitions). With *unequal error protection* (UEP) before transmission, the partitions with higher importance can be protected better, for example by using stronger FEC codes to protect the high priority NALUs. However, a loss of a header partition (A partition) will render the related B and C partitions useless. This is why the use of data partitioning is advisable only if very strong protection can be afforded to protect the A partitions. For example, it is reported in [1] that FMO outperforms data partitioning even when the A partitions are protected by sending each of them two or three times in different transport packets.

2.2 Forward Error Correction

There are several different alternatives to implement FEC for video streaming applications. The simplest method is to repeat the source data blocks or packets several times to increase the probability that at least one of the redundant copies is received. This method may introduce enormous overhead and often much smaller amount of redundancy can produce comparably good results. A well-known and simple technique is to apply binary exclusive OR (XOR) operation across the source symbols. If one of the source symbols (packets) is lost, it can be recovered by applying the XOR operation to the FEC symbol and the remaining source symbols. RFC 2733 lists several ways of using XOR-based FEC schemes [6].

Optimal error correcting capability can be achieved by using *maximum distance separable* (MDS) codes, such as Reed-Solomon (RS) codes. An RS(n,k) encoder takes k source symbols as input and generates n-k FEC symbols as output. The group of n symbols (k source symbols and n-k FEC symbols) is defined as a *coding block* that is transmitted over a lossy transport channel. An RS decoder can then regenerate the original source symbols from any k of the n symbols in the block [7]. Unfortunately, if more than n-k symbols are lost, none of the lost source symbols can be recovered. This is a clear weakness in case the fraction of lost packets even occasionally exceeds n-k symbols per block. Streaming applications are likely to experience bursts of losses, which may result in dramatic quality drop when several large packets are lost consecutively. In this case, it would be better to recover at least part of the lost packets to avoid the loss of consecutive media packets.

Typically, interleaving can be used to address the problem of bursty packet losses, in order to attain a smooth distribution of the lost media units [8]. However, efficient interleaving across several RS coding blocks causes significant latency when coding blocks are long. To achieve smoother data recovery performance *within* a coding block it is possible to use partial RS codes [9] or *low-density parity check* (LDPC) codes optimized for high loss rates [10]. Unfortunately, the cost of better error recovery probability at high loss rates is the lower error recovery probability at low loss rates. In addition, partial RS codes leave part of the data unprotected, and LDPC codes cannot achieve as good overall error recovery capacity as RS codes. This is why these approaches only offer a limited flexibility in the design of the error control. This is certainly not ideal for common applications where data are not clearly distributed into different levels of importance.

In order to design more adaptive error control solutions, UEP scheme based on *priority encoding transmission* (PET) has been proposed in [11]. PET scheme segments the source data in units of different priorities and protects these units unevenly with different erasure correction codes (typically, RS codes with different code rates). Then, data units from each priority class are allocated in each packet as illustrated in Figure 1. In this example, layer 1 (highest priority) is protected with a strong RS(7,2) code, layer 2 with RS(7,4) code and layer 3 with RS(7,6) code. With unequal error protection, the data of highest priority are more likely to be fully recovered. At the same time, the number of quality levels that can be decoded increases with the number of packets received by the decoder.

Recall that RS codes can only recover data when the number of losses is smaller than *n-k* packets. Since packets can contain source media data or FEC redundancy, the residual data loss rate does not only depend on the fraction of lost units in a coding block, but also on *which* units are lost. For the same number of lost packets *L>n-k*, the data loss process might be very different, depending if the lost packets cover *L* FEC packets, *L* media data packets, or a mix of media data and FEC packets. This leads to varying loss recovery performance, and unstable quality at the decoder.

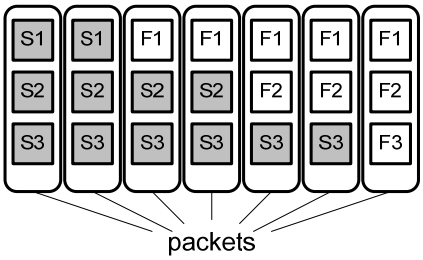

Figure 1. Priority encoding transmission. S=Source, F=FEC unit, numbers denote different coding blocks (layers).

To alleviate the stability problem and make the residual data loss characteristics smoother, a packetization strategy for short audio frames has been proposed in [12]. Similarly to PET, the proposed scheme allocates several data units in each packet. However, the source data units are built to be approximately equal in importance. Therefore, equal FEC protection is implemented. The target of this design is to spread source and FEC units evenly among packets. In addition, every coding group should occupy a unique set of packets. The scheme is illustrated in Figure 2 (in this example, an MDS(3,2) code is used for all 7 coding blocks). More details of the scheme are explained in Section 3.

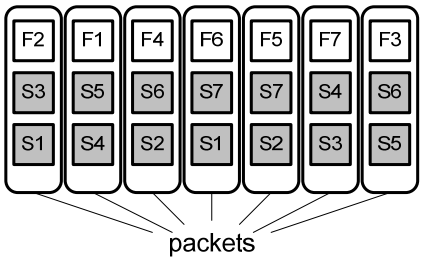

Figure 2. Data unit allocation as proposed in [12]. S=Source, F=FEC unit, numbers denote different coding blocks.

3. PROPOSED SCHEME

In this paper, we focus on a simple streaming scenario, where H.264/AVC video is transmitted over a packet erasure channel using IP/UDP/RTP protocol stack without retransmissions. In order to improve the performance of error concealment, we have adapted the packetization scheme proposed in [12] for streaming H.264/AVC video. Allocation of data units is said to be *ideal* if there is no *conflict* between any two (or more) coding blocks. In this context, a *data unit* may refer to either a source data unit (such as NALU) or FEC data unit. Conflict of coding blocks is defined as follows: we say that two blocks conflict if they occupy two or more same packets. For example, if two RS(5,3) coding blocks occupy packets {1,2,3,4,5} and {1,2,6,7,8}, respectively, they are in conflict, since they both occupy packets 1 and 2. In contrast, coding blocks that occupy packets {1,2,3,4,5} and {1,6,7,8,9}, are not in conflict.

The benefit of ideal allocation is that the fluctuation of the residual loss rate can be reduced as small as possible when packet losses hit source NALUs and FEC units unevenly. Since every possible combination of packet losses lead to different mix of data unit losses within each non-conflicting coding block, the residual NALU loss rate converges toward average with all possible combinations of a certain number of lost packets. Ideal allocation can be achieved for an RS(*n*,*k*) code if the constraints in Equation (1) are fulfilled (*B* coding blocks are allocated in *P* packets, each packet can accommodate *N* units). Large values of *n* and *B* would result in smoother distribution of residual losses, but on the other hand, longer latency due to longer packetization cycle. Often, it is simplest to choose *N*=*n* and *B*=*P*, and then *P* is easily solved from Equation (2).

$$\begin{aligned} Bn &= PN \\ B(n^2 - n) &= P^2 - P \end{aligned} \tag{1}$$

$$P = B = (n^2 - n) + 1 \tag{2}$$

The algorithm (1) shown below (similar as in [12]) can be used to derive *B* unique combinations of *n* packets (among *P* packets in total, indexed from 1 to *P*) that do not conflict with each other. The computational overhead of the algorithm is not a significant problem, since it is possible to generate a precomputed table of ideal packet combinations offline. Function `find_ideal_comb` is called recursively to test exhaustively all possible combinations, starting from {1,2,…,*n*}, until all *B* non-conflicting combinations have been found (or all possible combinations have been trialed and the algorithm fails). When the ideal allocation is known, the packetizer takes *B* RS coding blocks for each packetization cycle and allocates the NALUs and FEC units among packets according to the attained list of non-conflicting combinations.

Algorithm 1: find the optimal allocation of data units

```
call find_ideal_comb( {Ø}, {1,2,…,n} )

function find_ideal_comb(comb_list, test_comb)
  insert test_comb to comb_list
  if size(comb_list) == B
    return true
  Endif
  while next_test_comb != {P-n+1,…,P}  // last comb.
    compute next_test_comb
    if next_test_comb does not conflict with comb_list
      if find_ideal_comb(comb_list, next_test_comb)==true
        return true
      else
        remove next_test_comb from comb_list
      endif
    endif
  end while
  return false
end function
```

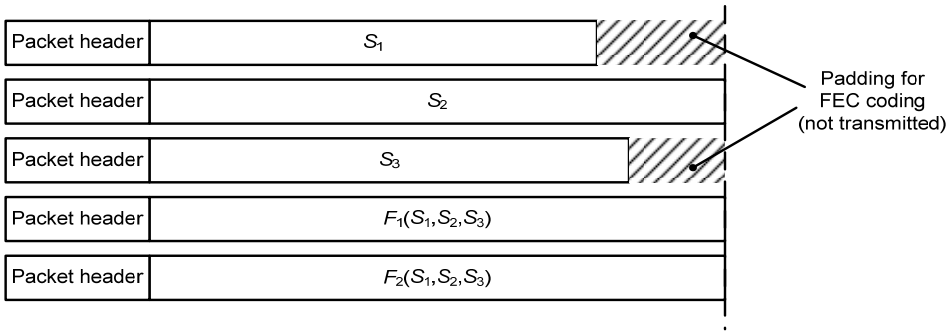

Figure 3. Baseline packetization and FEC.

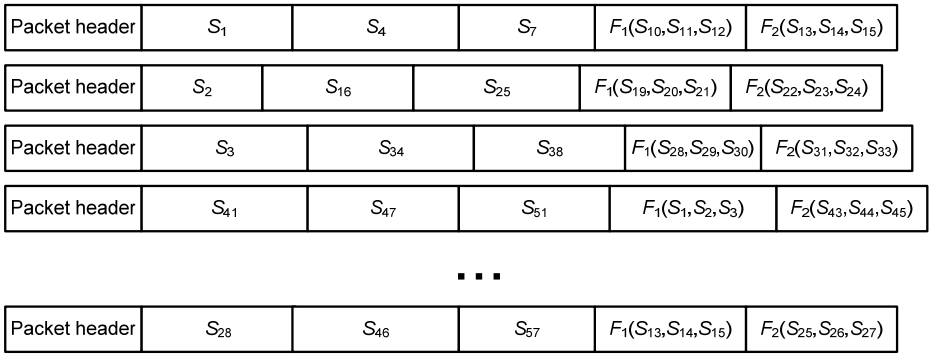

Figure 4. The basic principle of the proposed combined packetization and FEC.

In [12], it is assumed that the size of all the source data units and FEC units is the same. Unfortunately, this assumption is not realistic with many advanced coding standards. In the JM reference encoder for H.264/AVC [13], the maximum NALU size can be defined, but NALU sizes cannot be made exactly the same (in theory, it would be possible to get close to a constant size, but it would pose a significant extra complexity burden at the encoder). In RS coding, the FEC units must be as long as the longest of the source units (see Figure 3); this is why the actual FEC overhead in bytes is usually larger than the FEC overhead measured in the number of packets $(n\text{-}k)/k$. For example, if three packets of sizes {150,200,250} bytes are protected with two FEC packets of 250 bytes, the overhead in the number of packets is 2/3, but the actual overhead in bytes would be $(2\cdot250)/(150+200+250)=5/6$. In order to minimize the extra FEC overhead, the length of the NALUs should fluctuate as little as possible. One of the benefits of using small NALUs is that the variation of NALU sizes can be suppressed.

To minimize the transport packet header overhead, the total packet payload should be close to the *maximum transport unit* (MTU) size (in the Internet, MTU is usually assumed to be around 1400-1500 bytes). Therefore, when there are N units packed in each packet, the maximum NALU size should be close to MTU/N. However, the maximum NALU size cannot be chosen arbitrarily to fulfill this condition, since the meaningful range of NALU sizes depends also on the coding parameters. As a rule of thumb, the lower the bitrate, the smaller the NALUs should be. On the other hand, very small NALU size cannot be recommended since it would decrease the coding efficiency due to increased NALU header overhead.

4. EVALUATION

We show the importance of efficient packetization in error control by comparing the proposed scheme to a baseline system represented in Figure 3. RS(5,3) code has been chosen for erasure protection, ie. each group G_i of three source NALUs ($G_i=\{S_{3i},S_{3i+1},S_{3i+2}\}$) is protected by two RS FEC units ($F_1(S_{3i},S_{3i+1},S_{3i+2})$ and $F_2(S_{3i},S_{3i+1},S_{3i+2})$). This code is considered as a good option for streaming applications, since the coding block is rather short and the protection level is sufficient for relatively high loss rates. In fact, even lower protection level could be sufficient in many practical networking scenarios, and our example is primarily targeted on environments where high packet loss rates may occur (such as some wireless multicast systems, for example).

For the baseline scheme, we have used maximum NALU size of 1400 bytes. One NALU or FEC unit is allocated in each packet. Source packets and the respective FEC packets are transmitted consecutively as illustrated in Figure 3. In the concept system, the maximum NALU size is 280 bytes and five data units are accommodated per packet (three NALUs and two FEC units). Each packetization cycle contains 63 NALUs and 42 FEC units, resulting in 21 packets per cycle in total. With these parameters, ideal allocation of data units can be attained. The packetization scheme is illustrated in Figure 4.

In order to evaluate the performance of the proposed scheme, we have simulated both baseline and concept packetization schemes in Matlab. In the simulations, we have used two different CIF sequences ‘Soccer’ and ‘Foreman’, 30 frames per second. JM

reference codec version 13.2 [13] has been used for encoding and decoding the sequences. In the first set of experiments, both sequences have been encoded using the two different maximum NALU sizes (280 and 1400 bytes) and quantization parameter (QP) 30. Every ninth frame is an I-frame, B-frames are not used. The standard JM concealment (motion copy) has been used and FMO (dispersed mode) has been enabled to facilitate error concealment. Because the original sequences were rather short, several copies were concatenated to form longer sequences of approximately 600 frames to ensure that the experiment results are statistically significant. This configuration represents a typical video sequence suitable for streaming.

Because of the larger NALU header overhead, the bitrate of the encoded sequences is slightly increased when small NALUs are used. However, it is worth noting that with large maximum NALU size the variation of actual NALU sizes is also higher. This leads to larger FEC overhead and smaller average packet size, which in turn increases the average packet header overhead. Tables 1 and 2 summarize the essential parameters of the 'Foreman' and 'Soccer' sequences, respectively. As the results show, the approximately 5% higher bitrate of the stream encoded with small NALUs is largely compensated in transport rate due to lower FEC and header overhead (typical IP/UDP/RTP headers of 40 bytes are assumed to form the packet header overhead). The difference in transport stream bitrate is approximately 2% only.

Packet erasure channel have been simulated by dropping packets randomly in an independent fashion. In practice, this is done by generating random numbers r_i between 0 and 1, and dropping packet i if r_i is smaller than the threshold value θ ($0<\theta<1$). Six different values of θ={0.075, 0.15, 0.2, 0.25, 0.3, 0.35} have been used to cover the range of meaningful packet loss rates for each four test sequences. According to some studies, a realistic packet loss scenario involves bursty packet erasure patterns [14,15]. However, bursty packet losses can be spread more smoothly by interleaving, and in fact, our scheme scrambles the original sequence of NALUs quite efficiently. A random packet loss model is therefore considered appropriate for the evaluation of the proposed system.

Table 1. Characteristics of the 'Foreman' sequences (QP=30)

	Long NALUs	Short NALUs
PSNR	38.43	38.43
Average coded bitrate	624 kbit/s	653 kbit/s
Average packet size	1013 B	1332 B
Average transport rate	1,151 kbit/s	1,168 kb/s

Table 2. Characteristics of the 'Soccer' sequences (QP=30)

	Long NALUs	Short NALUs
PSNR	38.12	38.13
Average coded bitrate	791 kbit/s	832 kbit/s
Average packet size	1210 B	1341 B
Average transport rate	1,429 kbit/s	1,472 kbit/s

The resulting packet loss traces have been applied to real H.264/AVC streams and the sequences have been decoded, using the standard JM error concealment feature (motion copy). The results were analyzed by measuring the PSNR compared to the original video sequences. The resulting PSNR values have been plotted as a function of the observed packet loss rate in Figure 5 ('Foreman' sequence) and Figure 6 ('Soccer' sequence). As both of the curves show, the use of short NALUs and the proposed packetization scheme improve the video quality notably (approximately 1 dB), when the packet loss rate exceeds 0.15. At low packet loss rates below 0.1, the residual NALU loss rate is so low that the perceived quality degradation is negligible. The performance difference is rather similar for both 'Foreman' and 'Soccer', so it is assumed that the results can be generalized for a relatively large variety of different content types.

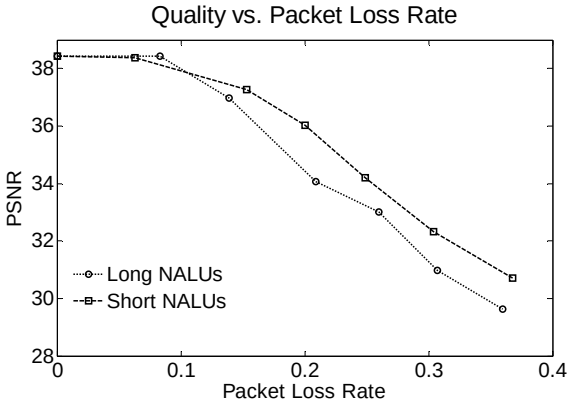

Figure 5. PSNR results for 'Foreman' sequence (QP=30).

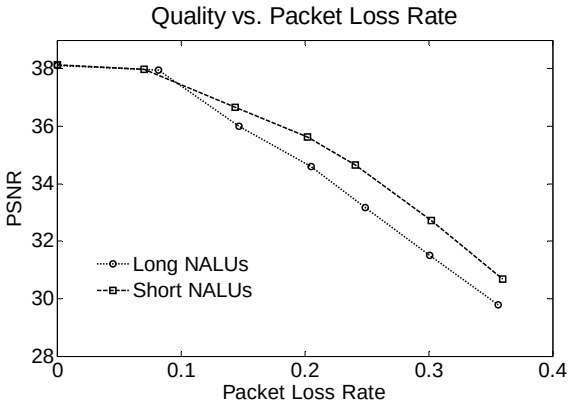

Figure 6. PSNR results for 'Soccer' sequence (QP=30).

Several studies suggest that occasionally occurring severe errors are more harmful for the overall subjective video quality than more smoothly distributed smaller errors [8,14,15]. With this in mind, we have analyzed also the variance of the PSNR values of individual frames in each test case. Since the proposed packetization scheme divides each frame in larger number of slices than the baseline scheme, the proportion of frames impacted by losses is higher. On the other hand, the impact of losing a small slice is smaller than the impact of losing a large slice. This is why the proposed scheme is supposed to reduce the quality fluctuation significantly. To illustrate this effect, Figure 7 shows a trace of PSNR values from 'Soccer' sequence with packet loss rate 0.2. Frames from about 100 to 175 show slightly better quality for the baseline scheme (long NALUs), but on the other hand, the large quality fluctuation observed in frames from 175 to 250 is significantly reduced by using the proposed scheme.

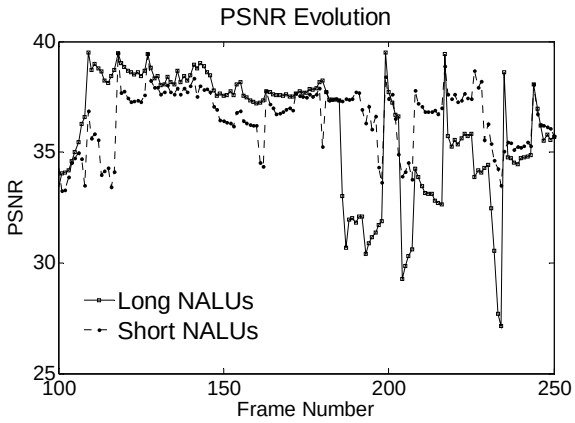

Figure 7. Trace of PSNR values in 'Soccer' sequence (PLR=0.2).

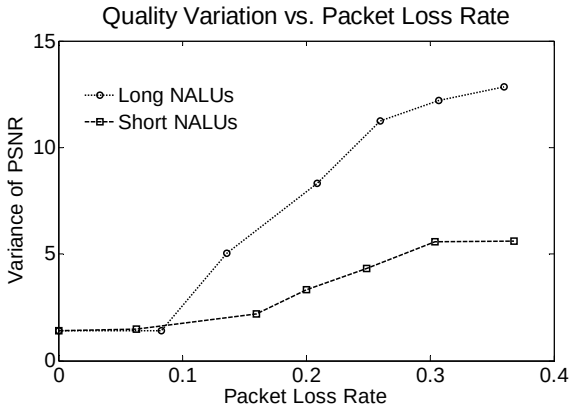

Figure 8. PSNR variance in 'Foreman' sequence (QP=30).

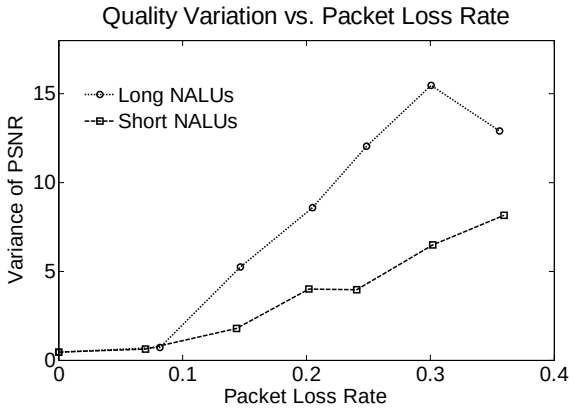

Figure 9. PSNR variance in 'Soccer' sequence (QP=30).

In order to analyze the quality fluctuation more systematically, we have measured the variance of PSNR values in each test case. The results are shown in Figures 8 and 9 for 'Foreman' and 'Soccer', respectively. As the results indicate, the use of small NALUs and the proposed packetization scheme reduce the observed variance in quality substantially at all packet loss rates higher than 0.1. This is why we could assume that the subjective quality improvement achieved by using the proposed scheme could be even bigger than the PSNR values shown in Figures 5 and 6 suggest.

In the scenarios described above, the average encoded frame size is approximately from 2500 bytes ('Foreman') to 3500 bytes ('Soccer'). Since FMO is used, each frame is divided in at least two slices (NALUs). Therefore, the average NALU size without restricting it at the encoder would be approximately 1250-1750 bytes. This justifies the maximum NALU sizes used in our experiments: maximum NALU size of 1400 bytes means typically two slices per frame, whereas maximum NALU size of 280 would result in approximately 10 slices per frame. However, with lower or higher bitrates these parameters would not necessarily be appropriate. In order to analyze the concept with different bitrates, we have repeated the experiments also with very high (QP=20) and very low (QP=40) quality versions of the 'Foreman' sequence.

The bitrate for the high quality stream is approximately 2.5 Mbit/s. Due to the MTU limitation in the traditional Internet, it is not reasonable to use NALUs larger than 1400 bytes. Therefore, we have used similar parameters as in the first set of experiments (maximum of 1400 bytes for large NALUs and 280 bytes for small NALUs), resulting in average of 8 slices per frame with large NALUs and 40 slices per frame with small NALUs. It is expected that the error concealment performance approaches perfect recovery asymptotically when the slices get smaller. This is why the difference in loss resilience assumedly suppresses when the slices get smaller, even though the relative difference remains the same as in our first test case. This assumption is confirmed by the experimental results shown in Figure 10. In fact, long NALUs seem to work slightly better at small PLRs (<0.2). The quality variation show more favorable results for small NALUs, but even then the difference is slighter than shown in Figures 8 and 9 above. Due to the limited space, the quality variance curves for this test case are omitted in this paper.

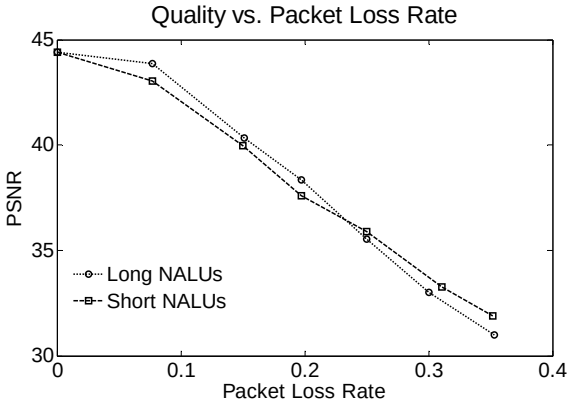

Figure 10. PSNR results in 'Foreman' sequence (QP=20).

In the low quality scenario, the bitrate for the encoded stream without NALU size restrictions is approximately 230 kbit/s, and the average frame size is about 1000 bytes. Therefore, a reasonable maximum NALU size with FMO would be 500 bytes.

We have repeated similar experiments as described above with the low quality bitstream, using maximum NALU size of 500 bytes for large NALUs and 100 bytes for small NALUs. These sizes are roughly in the same proportion to the bitrate as in the first set of experiments with QP=30. The resulting PSNR value curves are shown in Figure 11. As the results show, the performance difference between small and large NALUs is resembles relatively accurately to the behavior observed in the first set of experiments. The PSNR variance curves, although not presented here, show similar tendencies.

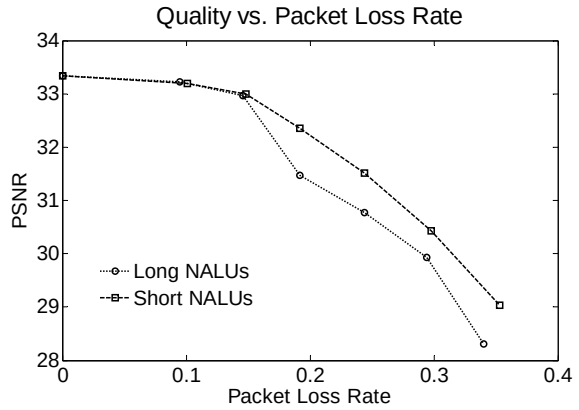

Figure 11. PSNR results in low quality 'Foreman' sequence (QP=40).

As the results show, performance of error concealment can be substantially improved by dividing video frames in small slices and using a sophisticated scheme for packetizing and protecting the slices. However, to ensure optimal performance, the size of NALUs should be selected carefully. If the slices are too small, the benefit of the scheme is lost due to the decreased compression efficiency. On the other hand, MTU of the underlying network defines the upper limit for NALU size. This is why the proposed scheme is not ideal for streaming video data of extremely high or low bitrates. Anyways, the advantages of the concept are obvious at typical bitrates in video streaming applications. Reasonable NALU size can be defined by simple heuristics, and deeper analysis of NALU size selection is out of the scope of this paper.

5. CONCLUSIONS

In this paper, we have proposed a packetization and FEC scheme for streaming H.264/AVC video, based on small individually decodable units (NALUs) and even distribution of source NALUs and FEC data units within transport packets. According to the experiments, the proposed scheme provides significant improvement in the subjective video quality in the presence of packet losses, both by increasing the overall quality and reducing the fluctuation of quality from frame to frame. The drawback of using small NALUs is the decreased compression efficiency caused by larger total NALU overhead. However, we have shown that due to the generally less varying size of small NALUs, the decreased compression efficiency can be largely compensated by reduced FEC and packet header overhead.

6. REFERENCES

[1] Wenger, S. 2003. H.264/AVC over IP. IEEE Trans. on Circuits and Systems for Video Techn. 13, 7 (Jul. 2003). 645-656.

[2] ITU-T, 2005. Advanced Video Coding for Audiovisual Services. ITU-T Recommendation H.264.

[3] Clark, D., and Tennenhouse, D. 1990. Architectural Considerations for a New Generation of Protocols. In Proc. of ACM SIGCOMM Symposium (Philadelphia, PA, USA, Sep. 24-27, 1990). 200-208.

[4] Wang, Y-K., Hannuksela, M., Varsa, V., Hourunranta, A., and Gabbouj, M. 2002. Error Concealment Feature in the H.26L Test Model. In Proc. of ICIP '02 (Rochester, NY, USA, Sep. 22-25, 2002). II-729-II-732.

[5] Yim, C., Kim, W., and Lim, H. 2005. Analysis and Performance Evaluation of Flexible Macroblock Ordering for H.264 Video Transmission over Packet-Lossy Networks. In Proc. of PCM '05 (Jeju Island, Korea, Nov. 13-16, 2005). LNCS 3767, Springer-Verlag, Berlin, Germany. 120-131.

[6] Rosenberg, J., and Schultzrinne, H. 1999. An RTP Payload Format for Generic Forward Error Correction. IETF RFC 2733.

[7] Lin S., and Costello, D. J., 1983. Error Control Coding: Fundamentals and Applications. Prentice-Hall, Englewood Cliffs, NJ, USA.

[8] M. Claypool, and Zhu., Y. 2003. Using Interleaving to Ameliorate the Effects of Packet Loss in a Video Stream. In Proc. of MNSA '03 (Providence, RI, USA, May 19-12, 2003). 508-513.

[9] S. Karande, and H. Radha. 2003. Partial Reed-Solomon Codes for Erasure Channels. Proc. of ITW '03 (Paris, France, Mar. 31-Apr. 04, 2003). 82-85.

[10] A. Dimakis, J. Wang, and K. Ramchandran. 2007. Unequal Growth Codes: Intermediate Performance and Unequal Error Protection for Video Streaming. In Proc. of MMSP '07, (Chania, Greece, Oct. 1-3, 2007). 107-110.

[11] Albanese, A., Blömer, J., Edmonds, J., Luby, M., and Sudan, M. Priority Encoding Transmission. In Proc. of FOCS '94 (Santa Fe, NM, USA, Nov. 20-22, 1994). 604-612.

[12] Korhonen, J., Huang, Y., and Wang, Y. 2006. Generic Forward Error Correction of Short Frames for IP Streaming Applications. Multimedia Tools and Applications 29, 3 (Jun. 2006). 305-323.

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

[14] Girod, B., Stuhlmüller, K, Link, M., and Horn, U. 1999. Loss Resilient Internet Video Streaming. In Proc. of VCIP '99 (San Jose, CA, USA, Jan. 1999). 833-844.

[15] Huitika, T., and Driessen, P. 2003. Datagram Loss Model for Non-Interactive Real Time Streaming Video. In Proc. of PACRIM '03 (Victoria, Canada, Aug. 28-30, 2003). 756-759.