ITU - Telecommunications Standardization Sector
STUDY GROUP 16
Video Coding Experts Group (Question 15)

Tenth meeting: Osaka, 16-18 May, 2000

Document Q15-J-53
Filename: q15j53.doc
Generated: 9 May '00

Question:	Q.15/SG16

Source: Stephan Wenger
TU Berlin and TELES AG
Sekr. FR 6-3
Franklinstr. 28-29
D-10587 Berlin
Germany
Tel: +49-172-3000813
Fax: +49-30-31425156
Email: stewe@cs.tu-berlin.de

Title: H.26L Error Resilience Experiments: First Results

Purpose: Information

1 Introduction

In Red Bank a breakout group discussed some concepts for a high level syntax for the H.26L project. A very informal description was made available as a result of this meeting in the Red Bank output document Q15-I-56R1. This syntax features data partitioning as its main error resilience tool.

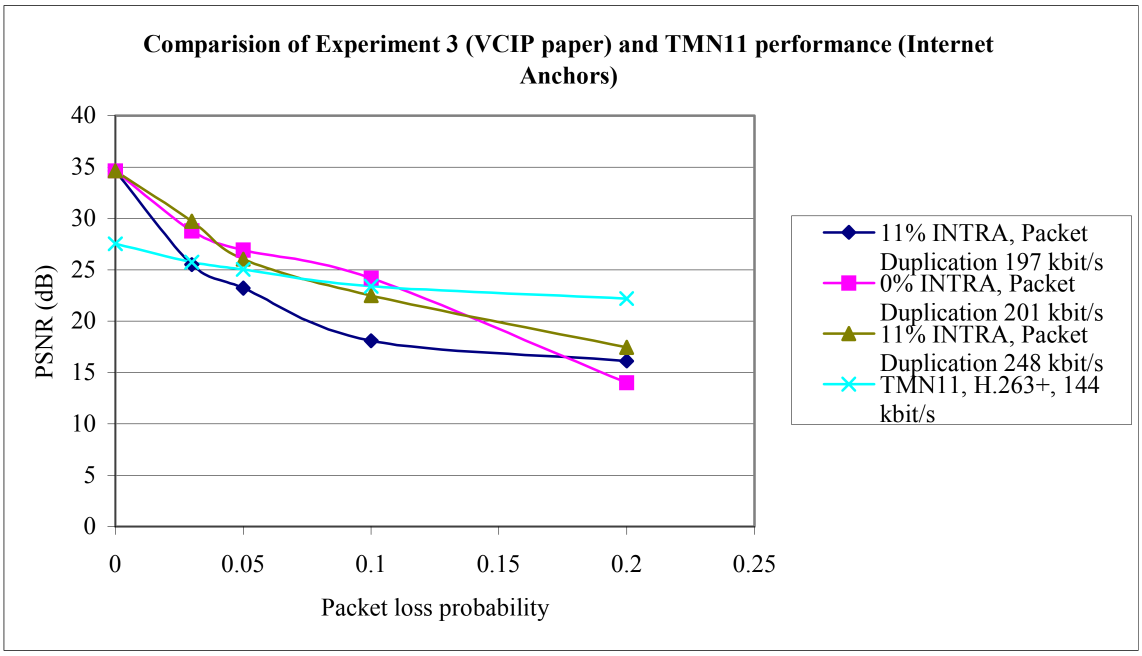
Since then, a group at TU Berlin implemented this scheme to show its feasibility and efficiency. This implementation work was performed based on Telenor's TML-1 software. Although since then siginificant improvements have been made, we believe that the results reported in the following are applicable to TML-3 and future improvements based on the Telenor proposal as well.

The results of simulations showed that the proposed scheme shows very good results when compared to the use of INTRA macroblock refresh as the only other available error resilience tool of TML-1. Of course, the performance gain would be far less dramatically when TML-1 would already include slices, so that mechanisms such as the one employed in TMN11 (for H.263 Internet transmissions) could be used. Still, even compared to such an environment (which could be called 'best currently known practise') the results of the data partoitioning scheme are still acceptable.

The following section of this paper contains a quick comparision of the performance of TML-1 w/ data partitioning compared to TMN11. The attached VCIP2000 conference paper provides more information on implementation details. Some simulations are also available in the accompanying MPEG file (12 MB size). See the conference paper for a detailed description of the contents.

2 TML-1 w/ DP compared to TMN11 (Internet case)

As mentioned before this comparision is not very fair, because TMN11 makes extensive use of slices whereas TML-1 does not have this concept. Furthermore, TMN11 uses a fixed bitrate environment with rate control and an adaptive way to apply source coding-based error resilience (Intra MBs) whereas all this is done in a very static way in case of TML-1. Finally, the TML-1 based work was not performed according to the simulation conditions (mostly because Telenor's TML-1 software does not include rate control yet, and we were unable to implement RC due to time constraints).



The absolute PSNR values are not interesting, because coding bitrates are so different, and because of the differences in PSNR easurement, where the 7.5 fps Foreman sequence is TMN11 is penalized severly because of its uncoded frames. More important, however, is the amount of PSNR drop for each simulation as the packet loss rate increases. The better an algorithm, the less severe ishould the impact of a higher packet loss rate be.

Clearly, TMN11 still outperforms the data partitioned scheme significantly. The best tested TML-1 based scheme for PL rates up to 10%, using packet duplication for the motion vectors but no source coding error resilience, hshows a PSNR loss of more than 30% between 0% and 10% packet loss rate, whereas that loss in case of TMN11 is only around 12%. More observation like this can be easily made.

3 Conclusions

The attached VCIP paper shows that the DP scheme for TML1 generally works and significantly improves the picture quality. This cane also be observed using the supplied MPEG file. The, in many ways unfair, comparision with TMN11 shows, however, what a long way we still have to go to include error resilience into H.26L that achieves quality levels similar to TMN11/H.263+.

A High Level Syntax for H.26L: First Results

Stephan Wenger[*]
Technische Universität Berlin, Department of Computer Science

ABSTRACT

This paper introduces some preliminary results of the standardization process of ITU-T's H.26L project. This forthcoming video coding standard will not only significantly improve the coding efficiency, but it will also introduce new concepts such as network friendliness, which are not common in current video coding approaches. The paper focuses on the high level syntax that resides hierarchically above the macroblock layer. Data partitioning techniques are used to separate data of different types from each other. The partitions are arranged in packets of data with different importance for the reproduced picture quality. Along with unequal error protection, which can either be a function of the underlying network or implemented on the application layer, the error resilience and thus the reproduced picture quality in error prone environments is greatly improved. To verify the findings, simulation results for an Internet/RTP environment, based on real-world observations of the current Internet that do not assume network-based quality of service, are included.

Keywords: Video Coding, H.26L, Standard, RTP, Error Resilience, Data Partitioning

1. INTRODUCTION

To achieve high reproduced video quality in error prone, low latency applications a mixture of error resilient source and channel coding is often appropriate. For real-world systems, the source coder typically conforms with one of the various video coding standards, such as H.263[1]. These standards allow enhancing error resilience by the application of redundancy in the source coding, i.e. by picture segmentation or by coding non-predictive (INTRA) information. The basic syntax is, however, identical to non-error resilient coding. The channel coder, on the other hand, adds redundancy as well, but usually without, or with only very limited, knowledge of the data structures conveyed. Such media unaware channel coding cannot distinguish between more or less important information in the data stream.

One of the major design goals of the ITU-T SG16/Q15's newest video coding standardization project, known as H.26L, is good error resilience and network friendliness[2]. To achieve such goals, current proposals for H.26L include a high level syntax that make a radical departure from concepts traditionally associated with video coding, such as a bit stream or synchronization markers[3]. Instead, for each network architecture, an appropriate high level syntax is used. This allows the video coding algorithm, in addition to the traditional means of applying redundancy to achieve error resilience, to generate entities of data that should be conveyed with a well-defined service class. When no differentiated service network support is available, then the channel coder functionality can be used to match these demands as closely as practical.

In this paper, we first introduce the core codec algorithm of H.26L Test Model Long-Term #1 (TML-1) as it was defined in late 1999[4]. While, at time of publication, this test model description was already outdated, many of the key technologies are still identical. Then, we introduce the data-partitioning scheme. This is followed by an outline for a packetization scheme for IP/UDP/RTP-based systems, such as H.323 systems or the MBONE tools, and is discussed along with the necessary error concealment mechanism. Finally, some simulation results show the effectiveness of the introduced scheme.

2. THE TML-1 CORE ALGORITHM

The TML-1 core algorithm uses inter picture prediction, augmented by motion-compensation, and transform coding of the residual signal. In addition, it allows for transform coded intra information. These techniques are generally used in current video coders such as H.261, H.263, or by the various members of the MPEG family.

The source picture format is common with H.261. YUV 4:2:0 coded source pixels are arranged in macroblocks of 16 x 16 pixel. Each macroblock consists of 16 blocks that contain 4 x 4 pixel. These 4 x 4 pixel information, or the residual in case of inter coding, is transformed using an exact, DCT-like, integer transform. The use of the integer transform avoids rounding errors common for integer implementation of DCT-based codecs, and, therefore, allows for efficient implementation on integer processor architectures.

Seven different inter coding modes are possible, which differ in the number of motion vectors per macroblock, and the size and shape of the regions those motions vectors are applied to. Figure 1 illustrates the size and shape of such regions.

[*] Email: stewe@cs.tu-berlin.de; Fax: +49-30-314-25156; WWW: http://kbs.cs.tu-berlin.de/~stewe/vcip2000

The test model's coder uses a macroblock-based, rate-distortion optimized mode decision process to decide on which of the coding modes is to be used.

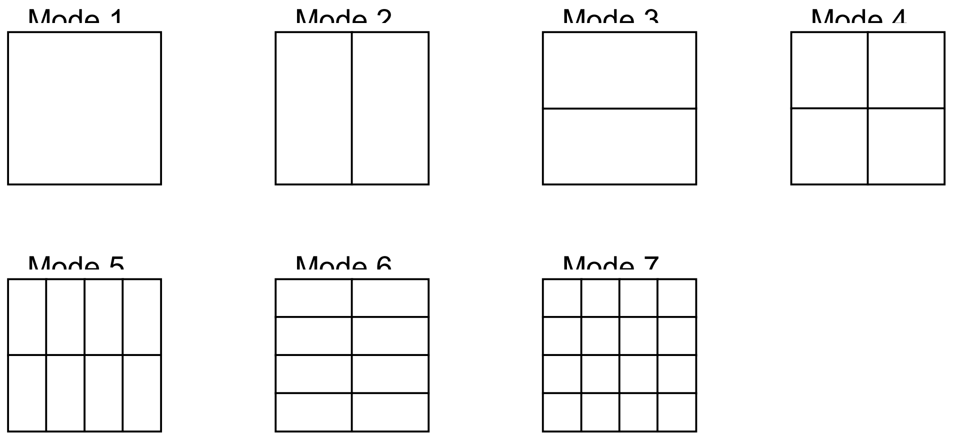


Figure 1: Spatial shape and size of regions within a macroblock for TML-1 motion vector coding

The accuracy of the motion vectors is currently subject to core experiments within Q.15/16 of the ITU-T. TML-1 proposed originally three different modes with an accuracy of ½, 1/3 and 1/6th of a pixel.

TML-1 allows for multiple reference frames. When using predictive coding, it is possible to decide for each macroblock to use the latest available, or an earlier reference picture for prediction. Such a technique leads to increase coding efficiency, but is also known as a valuable tool for error resilience purposes, as discussed in[5].

Intra coding is performed in the same way as in other video codecs. Several forms of intra prediction can be employed. The intra coder of TML-1 is certainly one of the less efficient mechanisms of TML-1, and it is likely that it will be changed significantly in the future For this reason and also because of its complexity, the intra coding is not discussed further.

The entropy coding employs exactly one regular variable length code table VLC. Any and all symbols generated during the earlier stages of the coding process are entropy coded according to this table. There are no additional fixed length codes, or special VLC tables for the various symbol types. Using one uniform VLC table has obvious advantages from an implementation point-of-view, but might also prove useful in bit error prone networks.

2.1. TML-1 Syntax

TML-1 does not contain any in-picture fragmentation mechanisms such as slices or GOBs, although such mechanisms are necessary both from a delay and from error resilience point-of-view, and, therefore, will be added in the future. Syntactically, a coded picture consists of some picture header information containing information relevant for all macroblocks of a picture and the coded macroblock information.

The TML-1 syntax diagram is depicted in

Figure 2. Each block in the diagram represents one VLC-coded symbol. Many of the symbols can be skipped, as announced by MB-Type and the content of the picture header. The following syntax elements are used:

- Picture header information, including the Temporal Reference (TR), the quantizer step size (PQP), and the picture type (intra/inter, Ptype).
- MB_Type: Valid macroblock types include skip, intra, or one of the already discussed inter coding modes that differ in the number of motion vectors and the size and shape of the regions covered by those vectors.
- Intra-pred-mode: Mode of Intra prediction, not discussed further here
- MACC: Motion vector accuracy
- Ref_frame: The temporal reference of the reference picture to be used for prediction.

- MVD: motion vector data. The number of motion vectors is announced by MB_Type. For each motion vector, two symbols are conveyed.
- CBP: Coded block pattern, similar to CBPs in H.261

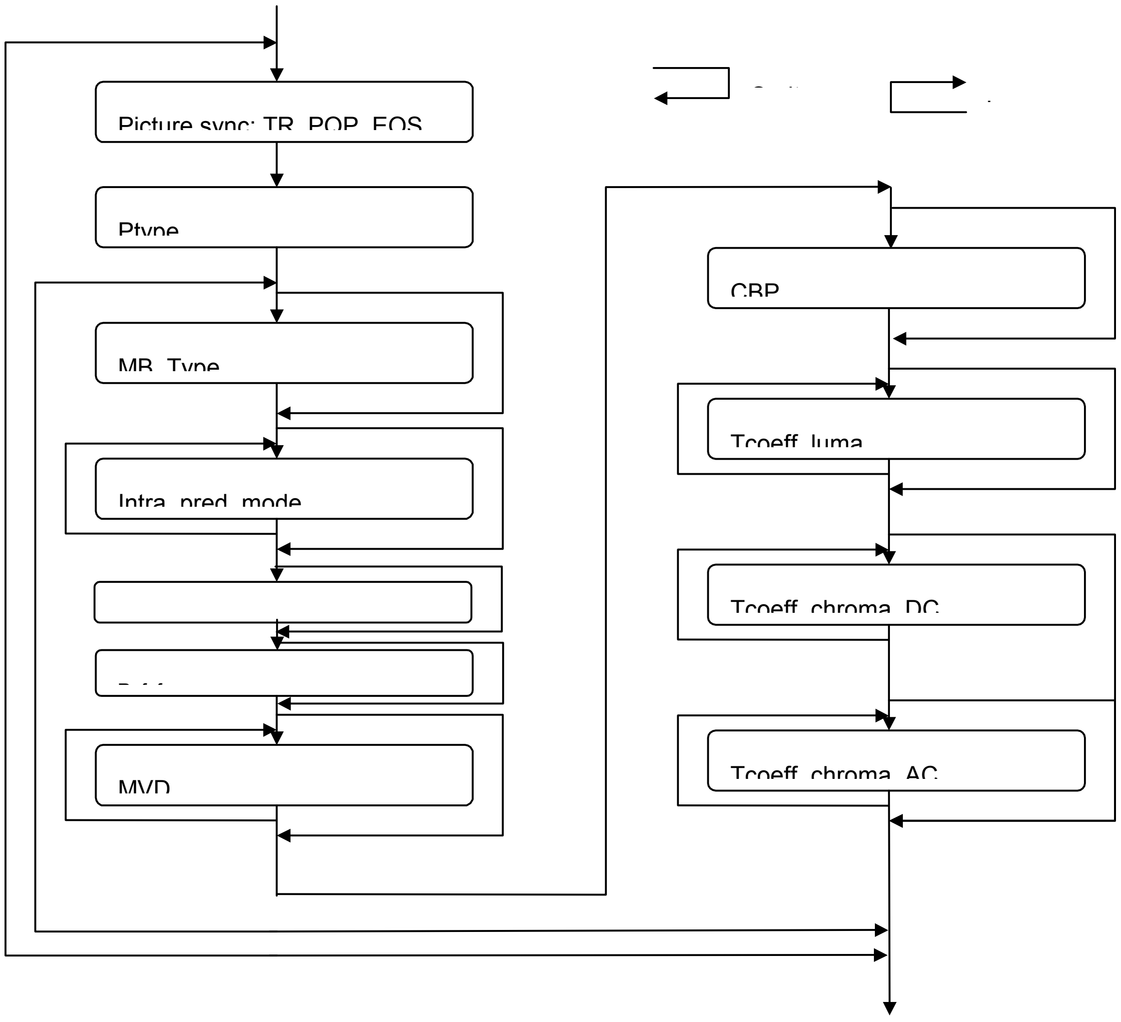


Figure 2: TML-1 Syntax Diagram

- Tcoeff_luma: Luminance transform coefficients. A run/level coding scheme is used, and the end of the transform coefficients for a certain block is announced by an EOB codeword, similar to H.261.
- Tcoeff_chroma_DC: DC chroma transform coefficients.
- Tcoeff_chroma_AC: AC chroma transform coefficients. The chroma transform coefficients use a different mapping of values to VLC symbols for AC and DC coefficients.

As a general rule, the types and numbers of all codewords for a coded macroblock are announced by previous codewords. The only exceptions to this rule are the transform coefficients, where an EOB symbol is used to determine the number of coefficients.

2.2. TML-1 Performance and computational complexity

A software implementation of a TML-1 compliant coder and decoder was made available by the proponent, and used to verify the efficiency of the coding algorithm. It was found that, for low bit rates and most sequences, bit rate savings of 50% or more compared to an H.263+ / TMN11 codec is achievable. Note that TMN11 uses many of the optional tools of H.263+, as well as a rate-distortion optimized mode decision process, and can be seen as the most efficient H.263+

implementation publicly available. Q15/16, therefore, believes that the gain in coding efficiency is mostly, if not completely, caused by the new features of TML-1, and not by a suboptimal implementation of H.263+.

The computational complexity of TML-1 is several times higher than the high complexity mode of TMN11. Coding a single QCIF picture, using the available, unoptimized software, takes up to 10 seconds on a PENTIUM-II – 400 system. This high complexity is mostly the result of the complex mode decision process and the use of five reference pictures. It is, however, believed that the complexity can be radically reduced without hurting the coding efficiency too much.

3. AN ERROR RESILIENT, HIGH LEVEL SYNTAX FOR TML1

3.1. History, Assumptions and Constraints

During the Red Bank meeting of Q15/16 in October 1999, a breakout group chaired by the author discussed an approach for a network friendly, error resilient high level syntax for TML1. At the starting point of our discussions it was observed that, when using TML-1 type – or any other common type – of video coding, there are certain dependencies of various data types. If, for example, the picture header with its picture type information is lost, then it is very difficult or even impossible to make use of any macroblock data, even if such data is undamaged. When the macroblock type information is missing, then it is impossible to decode that macroblock because it is unclear how the following symbols should be interpreted. It might even be possible to partly reconstruct a coded picture even if some less important parts are missing.

Data partitioning in the video coding is the appropriate tool to separate information of various data types from each other. Instead of having all symbols representing a coded macroblock concatenated together, symbols from all macroblock of a given data type are concatenated together and conveyed as a whole. That is, for example, that all MB_Type symbols for all macroblocks form a MB_Type partition. Similarly, there are partitions for all other syntax elements. This allows separating important from less important data, which in turn helps to appropriately apply unequal error protection schemes. The main disadvantage of data partitioning is the added delay on bandwidth-limited links, because the decoding of the first macroblock can only start once all partitions are received. On packet networks, however, this is much less an issue, because, typically, only complete packets are conveyed to the application, and those packets often have to be rather large (containing the whole picture) due to overhead/payload relationship constraints. We, therefore, decided that the high level syntax should make intensive use of data partitioning.

It was further observed that even under real-time constraints it is possible to abstract from bit error prone environments by applying some form of protocol support that ensures a bit error free, but packet lossy environment. Therefore, and due to time constraints, the breakout group considered only packet lossy environments. To facilitate the discussions we focused on an Internet IP/UDP/RTP environment, with its well known characteristics. These characteristics can be summarized as follows: a packet lossy environments with packet loss rates of up to 20% or more, no bit errors within packets, and packet sizes of up to 1500 bytes, which is the MTU size of the current Internet.

3.2. Low level source coding

In order not to interfere with the ongoing development work of the low level source coding we did not make any changes to those algorithms. We introduced, in particular, no loss-aware rate-distortion optimization mode decision or similar tools. All symbols and their VLC-coded representation are generated exactly as specified in TML-1. Only the arrangement of these symbols is changed.

3.3. Data partitioning

To facilitate implementation work and to be flexible in the future for different packetization schemes, the decoder generates an output file format, which contains the data partitioned, VLC coded symbols in a packet format. Each partition consists of a header containing information about type, size, and picture ID of the partition, and the partition data. This interim format is intended to be the input to the packetization process. The overhead is 12 bytes per partition plus typically 4 but maximal of 7 bits to achieve byte alignment of the VLC-coded symbols. The maximum number of partitions per picture is 10, as there are only 10 types of data (see section TML-1 Syntax above for the description of the data types). Therefore, the overhead per coded picture compared to original TML-1 syntax is 124 bytes. This corresponds to roughly 25% overhead for the realistically chosen picture size/bit rate scenario later used in our simulations.

3.4. Packetization

On the Internet, real-time media data is typically conveyed using an IP/UDP/RTP protocol hierarchy. The combined header overhead per packet of these three protocols is 40 bytes per packet. Conveying each data structures generated during the partitioning process in a single packet would lead to the doubling of the bit rate, which is unacceptable. Therefore, it is necessary to reduce the number of packets per picture, and, if possible, also the overhead introduced by the partition syntax.

We identified three different groups of data types, which form a hierarchy. In particular:

- Group 1: Picture Header, MBTYPE, IntraPred, Motion Vectors
- Group 2: CBPs, IntraCoeff

- Group 3: InterCoeff

For all three groups it holds true that all data of the numerically lower groups have to be available to decode data of the numerically higher groups. It is, for example, possible to use the motion vectors of group 1 along with the other information of that group, but ignore the missing group 2 coefficient data. Group 2 data, however, is needed to reconstruct group 3, as CBPs are necessary to reconstruct inter coefficients as well.

In order to facilitate implementation work, and as group 2 is typically very small, we for now ignored the potential additional benefits of using three groups, and settled for only two groups, each of which is conveyed in a single RTP packet:

- 'First' packet: contains Picture Header, MBTYPE, IntraPred, Motion Vectors
- 'Second' packet: contains CBPs and all coefficients

Using only two packets per picture reduces the packetization overhead to 80 bytes per picture.

To further reduce the overhead, the information concerning each partition that was generated by the partitioning process has to be reduced. To do so a 'part of partition' packet (POP-packet) is introduced. A POP-packet consists of a 16 bit header and partition data as indicated by the header. As there are only 10 different data types, we expressed the partition type information in 4 bits. In an RTP environment the picture ID can easily be reconstructed out of the RTP timestamp and, therefore, is not coded again. The remaining 12 bits of a 16 bit POP header field are used for the partition size information, measured in bits. This would allow for partitions up to 4096 bits, when using the value 0 as an indication for a size of 4096 bits. Since sometimes, especially when coding intra pictures, partitions can be bigger than 4096 bits, a value of 0 for the size also indicates that there is another POP packet of the same data type following somewhere in the packet.

The MTU size of the Internet is, due to historic reasons, roughly 1500 bytes, although all involved protocols theoretically allow for packet sizes up to 64 kbyte. In case of intra pictures, the size of all POP packets of the intra coefficients will often exceed the MTU size. We did not consider such a situation in our simulations, as there is a need for some picture segmentation mechanism on the source coding level, for example similar to MPEG slices, necessary in the future anyway, and that mechanisms could be employed to generate packets of reasonable size.

Figure 3 depicts the resulting packet structure assuming all partitions containing less then 4096 bits yielding one POP packet, except the Inter Coefficients partition which contains 6000 bits resulting in the need of two POP packets.

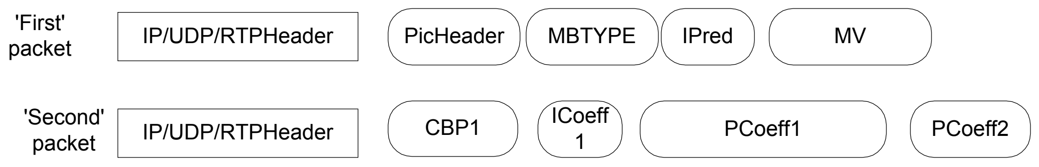


Figure 3: A coded picture consists of two packets, each of which contains POP packets of various data types

3.5. De-packetization

The de-packetization process is straightforward. If the received packet is a 'First' packet, which can be either identified by parsing the POP-header structure or by checking RTP's marker bit, then another packet is read. If that packet is the 'Second' packet of the same picture, which has to be checked using the RTP timestamp, then the partitions of both packets can be conveyed to the decoder. If the 'First' packet is missing, then the received 'Second' packet is not conveyed to the decoder, as the decoder will be unable to use any of the bits of that packet. If the second packet is missing, then only the 'First' packet is conveyed to the decoder. The decoder uses the unavailability of symbols of the second packet to trigger the use of error concealment.

3.6. Reconstruction and concealment

The reconstruction process follows exactly the algorithms defined in TML-1, except when the symbols of the second packets are missing. In this case, when the decoder requests a CBP information, a codeword provided by the bistream handling module that indicates that none of the blocks contains coefficients. This prevents the decoder from expecting coefficients that are due to the packet loss no more available. The low-level part of the decoder thus 'sees' only a picture that contains motion vectors, but no coefficients, in its inter macroblocks. While the loss of the residual signal certainly leads to picture degradation, that degradation is surprisingly low, as discussed in the simulation results.

4. SIMULATION RESULTS

To perform simulations to verify the efficiency of the concepts introduced above, we modified Telenor's TML-1 implementation to deliver the data partitioned VLC-coded symbols. A packetization tool converted this data into two RTP packets per picture, following the algorithm already introduced. Packet losses were applied using the same packet loss simulator and the same error patterns as used for the H.263 standardization work[6]. The resulting packet stream was converted back into the partitioned format using a de-packetizer, and reconstructed using the modified Telenor decoder. Modifications in the decoder included both the support for data partitioning, and the simple error concealment method introduced before.

Three sets of experiments were performed:

- Experiment 1 verified the functioning of the error concealment technique.
- Experiment 2 showed the impact of the use of redundant intra information during the source coding process.
- Experiment 3 finally added a channel coder with a very simple unequal error protection scheme to experiment 2, to better protect the packets containing motion vectors.

All three experiments were performed without rate control and using a fixed quantizer value, yielding coder-dictated variable bit rate. Real-world systems typically have to apply rate control to keep control over the generated bit rate. We believe, that the reported simplified results still have significance. Care has to be taken when interpreting the diagrams, as sometimes PSNR values for very different bit rates are compared in the same diagram.

All experiments were performed using 300 pictures of the QCIF-size sequence Foreman. The employed frame rate was 30 fps. A fixed quantizer value of 16 for the first intra picture, and 19 for all other pictures was used.

Although PSNR measurement is known to have poor match characteristics to the human visual sense, it is still the only widely accepted and freely available tool for objective image quality assessment. Therefore, luminance PSNR values, along with the bit rate, are used to report our findings. PSNR measurement is performed by comparing a reconstructed picture to all timely corresponding source pictures. That is, if pictures can not be reconstructed due to packet losses, the previous reproduced picture is used to calculate a PSNR value. This modified PSNR measurement is often used within Q.15/16 to penalize missing reconstructed pictures in lossy environments.

To allow for subjective quality assessment, the reconstructed video is available MPEG-encoded and in loss-less AVI format from http://kbs.cs.tu-berlin.de/~stewe/vcip2000.

Here we should make a final remark about bit rates. When working in a packet lossy environment, the bit rates at the receiver and the sender obviously differ due to the packet losses. In this paper, whenever bit rates are reported, they do not include any protocol overhead, and are measured at the sender. For a typical IP/UDP/RTP-based system, an additional overhead of 40 bytes per packet or 80 bytes per picture has to be added. At the fixed frame rate of 30 fps, this results in a packetization overhead of 19200 bits per second.

4.1. Experiment 1: Verification of the efficiency of the error concealment technique

In this simple experiment the encoder was configured not to generate any intra information, except where its own RD-optimization scheme suggested. That led to a coded bit stream that does not contain any additional redundant intra information to support error resilience. After packetization, either none, or the third, or the fourth, or both the third and the fourth packets were dropped. Note that the first two packets contain the first intra picture.

As the third packet contained necessary information to decode the fourth packet, we expected to see identical results when dropping both the third and the fourth, or only the third picture. When dropping only the fourth packet, the error concealment algorithm was able to produce higher reconstructed picture quality. The best quality was achieved by reconstructing data where no losses had occurred.

The resulting PSNR values are presented in Table 1. Clearly, the error concealment algorithm seems to work, and the resulting gains of almost 10 dB are as impressive as the total quality loss, when using concealment, of only 0.5 dB is.

Table 1: Performance of error concealment

Packets lost	PSNR (dB)
None	34.62
4 (which can be concealed)	34.13
3 (which cannot be concealed)	24.25
3 and 4	24.25

4.2. Experiment 2: Impact of Intra coding

In this experiment, the traditional means of redundant intra macroblock coding is employed in addition to the error concealment mechanism. Three source files were generated: without, with 5.5 % and with 11% intra coded macroblocks. As the intra macroblocks were coded at the same fixed quantizer value as the rest of the picture, the resulting QP without any losses was almost identical. The bit rate overhead, however, was substantial, as documented in Table 2:

Table 2: Intra macroblock refresh bit rate

Redundant intra macroblocks in picture	Bitrate (kbit/s)	% bitrate from optimal coding	PSNR (dB)
None	144	0 %	34.62
5.5%	172	20 %	34.62
11%	197	38 %	34.63

For those three input files, packet losses were applied and the resulting reconstructed PSNR was measured. The employed packet loss rates were 3%, 5%, 10%, and 20%. The packet loss patterns were obtained by experiments on the real-world Internet, and are used for all such experiments in Q15/16. Figure 4 plots the distortion against the packet loss rate for all three intra macroblock refresh rates.

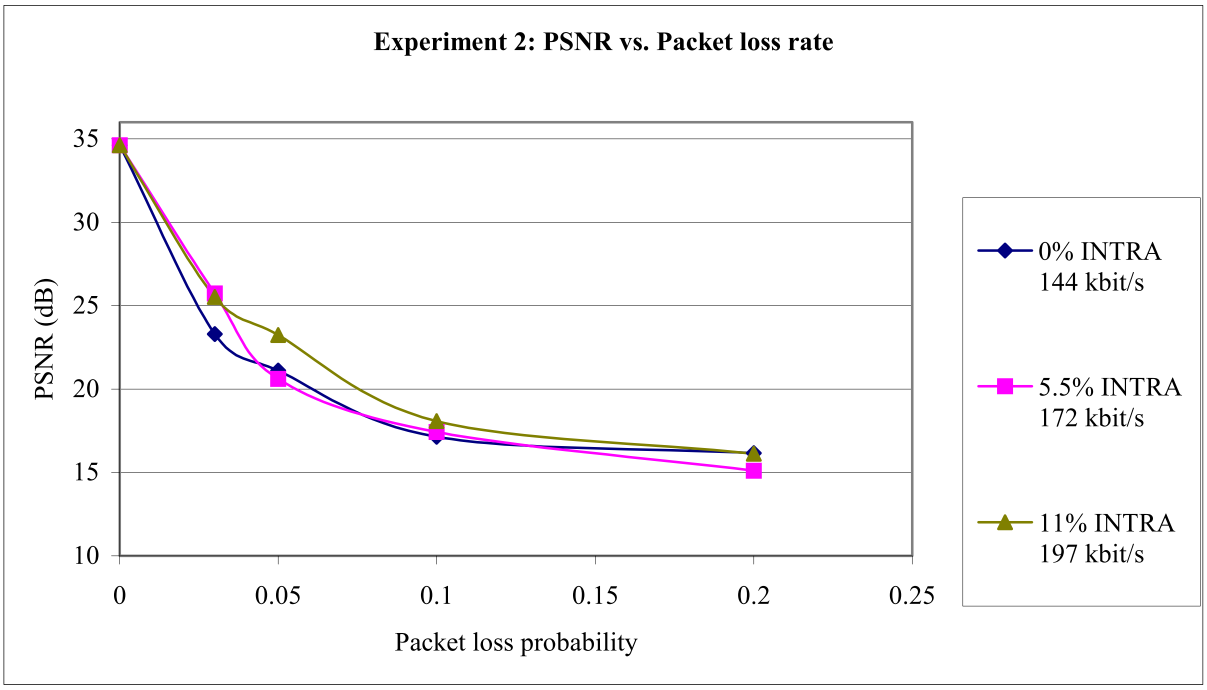


Figure 4: Efficiency of Intra macroblock refresh at various packet loss rates

Clearly, the results are disappointing. Even at the low packet loss rate of 3%, the PSNR drops by almost 10 dB from 34.6 down to 25 dB – a value that corresponds to a significantly worse reproduced picture quality with large annoying artifacts. Packet loss rates higher then 5% yield PSNR values below 20 dB, and the corresponding subjective quality is as devastating as those values suggest.

It is, however, observed, that intra macroblock refresh helps to improve the picture quality in packet lossy environments, especially at low packet loss rates.

Intra macroblock refresh has been shown to be a very useful tool to improve error resilience, and the amount of intra macroblocks used in our experiments are very similar to those we used in earlier, H.263-based, research[7], where the obtained results were significantly better. We, therefore, carefully checked our simulation environment for implementation errors, but couldn't find any. We believe, that two sources might contribute to these negative results:

- First, Intra macroblock refresh was performed by using a single line of intra coded macroblocks per picture, whereas our H.263 based research uses macroblocks that were distributed either randomly or by a loss-aware rate-distortion optimization algorithm. It might be possible that such algorithms would yield significantly better results. Time constraints prevented us to perform further experiments.
- Second, the intra coder of TML-1 produces a significantly higher relative overhead over inter coding then in H.263. That is, a typical size relationship of intra and inter macroblocks of TML-1 is smaller then in H.263, due to the much more efficient inter coding of TML-1. That leads to a very substantial bit rate increase for intra coded data, which prevented us from using an even higher intra macroblock refresh rate.

4.3. Experiment 3: Impact of a simple channel coder

It was observed in experiment 1 that error concealment is beneficial to the reconstructed picture quality. It was furthermore observed in experiment 2 that an appropriate number of redundant intra macroblocks also improved quality to a certain extend. In experiment 3, a very simple additional mechanism is introduced, which implements unequal error protection on the packetization layer.

It was noted earlier, that the 'first' packet of each picture, containing the motion vector and other header data, is necessary to interpret the 'second' picture that consists of CBPs and transform coefficients. It was further noted that a missing 'second' packet could be concealed by patching non-received CBPs to zero. Therefore, the 'first' packet is more important for the reconstruction process then the second picture and justifies additional bits for protection.

In experiment 3, all 'first' packets of a picture are transmitted twice. Such packet duplication is known to significantly improve reproduced media quality and, with some refinements, widely used for Internet telephony as Audio Redundancy Coding[8] or to protect the picture header of MPEG-4 or H.263 video data. At the de-packetizer, any duplicated 'first' packets are dropped, which, in real-world systems, could be easily implemented employing the timestamp and the marker bit of the RTP header.

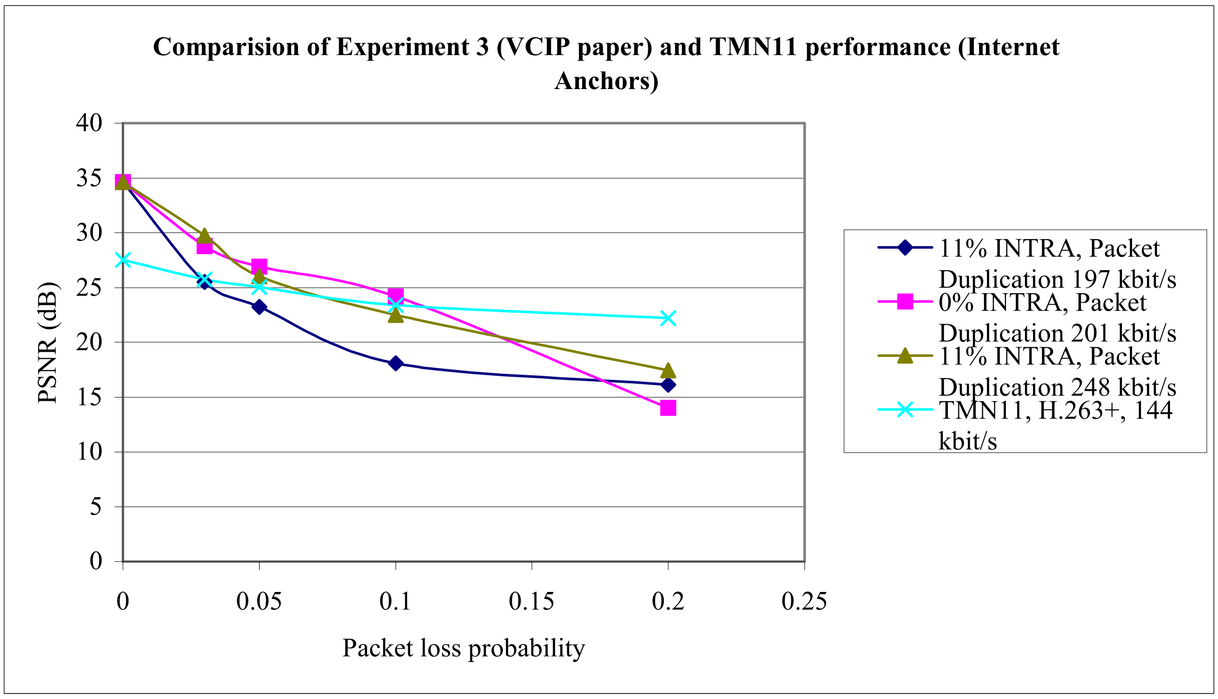


Figure 5: Performance of packet duplication

In Figure 5 we compare the best performing intra macroblock refresh algorithm of experiment 2 with header duplication. When comparing a source coding employing 11% Intra macroblocks for error resilience and a channel coding employing roughly the same amount of bits for packet duplication, the latter algorithm shows a performance gain of roughly 5 dB for all packet loss rates except 20%. When, spending bits for intra macroblock refresh, then the additional performance especially at higher error rates is even more impressive, and can be as high as 8 dB for 25% additional bit rate. We, therefore, believe that, at least at the moment, even a simple transport based algorithm like packet duplication is more efficient then the tested source coding based algorithm. This stands in contrast to our H.263-based research, where source coding based algorithms were more efficient. We believe that this is a result of the employed high level syntax.

5. FUTURE WORK

The discussed error resilient video transmission algorithm already shows significant improvements over the only built-in error resilience tool of TML-1, which is intra coding of macroblocks. Additional improvements are, however, possible and necessary. The following future research topics come to mind:

The current scheme processes the picture as a whole. While this might make sense for some low bit rate / high overhead scenarios such as dial-up connections to the Internet, high bit rate scenarios make forms of picture segmentation necessary, as well as environments that have comparatively low overhead and thus allow for small packet sizes. The author believes that a concept similar to MPEG's slices might be appropriate.

In this paper, a packet lossy environment that does not allow bit errors within packets was assumed. Clearly, all network environments including the so-called 'mobile', wireless networks, can be used in this manner by adding an appropriate protocol stack. Such an approach, for example, was followed for digital satellite TV. Many proponents of wireless interactive video phones believe that this approach might be the easiest and offers the best quality. Others, however, insist on the need of a video coding that can cope with bit error prone networks.

When assuming a bit error prone network, the use of the uniform VLC table would allow for new repair techniques. Alternatively, the use of more traditional means to cope with bit error prone environments such as RVLCs would also be possible, although the beauty of the current VLC design would then be lost. In both cases, the use of a data partitioning scheme similar to the one discussed here, and unequal error protection could further improve the picture quality.

ACKNOWLEDGEMENTS

The author wants to thank the members of the H.26L high level syntax breakout group of the RedBank meeting for the valuable discussions there: Thomas Stockhammer (Technische Universität München), Miska Hannuksela (Nokia Research Center), Michael Gallant (University of British Columbia), Matthieu Tisserand (UCLA), Jiangtao Wen (PacketVideo Corp), and Max Lutrell (PacketVideo Corp.).

REFERENCES

1. ITU-T Recommendation H.263, "Video coding for low bit rate communication", Feb. 1998
2. ITU-T SG16: "DRAFT Call for proposals for H.26L video coding" February 1998, available from ftp://standard.pictel.com/video-site/9801_Gen/H26LCfP.doc
3. ITU T Q.15/16: "A concept for a network friendly high layer syntax for H.26L", October 1999, available from ftp://standard.pictel.com/video-site/9910_Red/q15i56r1.doc
4. Bjontegaard, G.: "H.26L Test Model Long Term Number 1 (TML-1) draft 2", August 1999, available from ftp://standard.pictel.com/video-site/9908-Ber/Q15h36d2.doc
5. B. Girod, N. Färber: "Feedback-based error control for mobile video transmission", Proc. IEEE, Special Issue on Video for Mobile Multimedia, to appear, 1999
6. S. Wenger: "Packet loss simulation environment, version 2", October 1999, available from ftp://standard.pictel.com/video-site/9910_Red/q15i09.zip
7. S. Wenger, G. Côté: "Using RFC2429 and H.263+ at low to medium bit-rates for low-latency applications", electronic proceedings of the Packet Video Workshop 1999, New York
8. C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J. C. Bolot, A. Vega-Garcia, S. Fosse-Parisis: "RTP Pyaload for Redundant Audio Data", Internet RFC2198, available from ftp://ftp.isi.edu/in-notes/rfc2198.txt, 1997