When On-the-Fly Erasure Code Makes Late Video Decoding Happen

Pierre Ugo Tournoux
University of La Réunion, France
tournoux@gmail.com

Tuan Tran Thai
University of Toulouse, ISAE, TéSA, France
tuan.tran-thai@isae.fr

Emmanuel Lochin
University of Toulouse, ISAE, TéSA, France
emmanuel.lochin@isae.fr

Jérôme Lacan
University of Toulouse, ISAE, TéSA, France
jerome.lacan@isae.fr

ABSTRACT

This paper proposes "LD-Tetrys" (Late Decoding Tetrys), a solution based on an on-the-fly erasure code that attempts to solve the problem of late decoded packets usually considered as lost by the video decoder. LD-Tetrys has the following advantages: *i)* it drastically improves the trade-off between throughput and quality without modifying the codecs or adding complexity at the encoder side *ii)* it allows a simple but robust configuration. The only cost is a minor modification of the decoding process and a slight increase in the video decoding complexity. Last but not least, LD-Tetrys requires a much smaller playout buffer to obtain the same perceived video quality, bringing benefits for interactive applications.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications ApllicationsComputer conferencing, teleconferencing, and videoconferencing

Keywords

Network coding, video coding, real-time applications

1. INTRODUCTION

Video transmission dominates our current Internet traffic. As per Cisco forecast, consumer Internet video traffic will be 69 percent of all consumer Internet traffic in 2017, up from 57 percent[1] in 2012 [2]. The delivery of multimedia content is an active research area, especially for real-time and delay-sensitive applications which are main subject of RMCAT and WebRTC IETF research groups [3, 4]. The significant growth in video traffic can be explained by the deployment of high speed networks (e.g., 3G/LTE, fiber to home), mobile devices (e.g., laptop, smartphone, tablet) and advanced video codecs. The newly standardized video codec, the High Efficiency Video Coding (HEVC) [11], allows up to 50% encoding bit rate savings for an equivalent perceptual quality compared to the H.264/MPEG-4 Part 10 (also called H.264/Advanced Video Coding or H.264/AVC) which is enabled in most up to date end user devices. However, the higher compression efficiency leads the compressed video more sensitive to error/loss. Stephen Wenger shows in [15] that the Peak Signal to Noise Ratio (PSNR) decreases up to several dB when the loss rate is greater than 1%.

A simple solution to tackle the problem of video packet loss is to let the receiver detect the missing packets and request for the retransmission. This approach is used in TCP-based streaming solutions such as Adobe Flash Player and Microsoft Silverlight. It works fine if the playout delay is large enough (i.e., higher than 3/2 Round Trip Time (RTT)) so that the retransmitted packets arrives before the playout time of the frame it belongs. Unfortunately, the RTT is often too large compared to the delay the application can tolerate. In the context of low delay applications, the video decoder has to implement error concealment schemes in order to guess the value of the missing data [14]. To make the error concealment mechanisms effective, the video encoder may implement error resilience schemes such as Flexible Macroblock Ordering (FMO) with a cost of higher encoding bit rate than an encoded video without error resilience.

Another type of error resilience schemes frequently used falls in the family of application layer forward error correction (AL-FEC) and more particularly (n, k) block based erasure codes which generate n encoded packets out of k source packets and allow the recovery of the block if at least k packets among any of n encoded packets are received. The downside of these block-based erasure codes is that it requires complex probing of n and k as it trades off throughput and delay for residual loss rate. Recently, a novel erasure coding approach that prevents such complex configuration has been proposed in [12, 6, 13]. Tetrys [6, 13] that belongs to the class of on-the-fly codes, has the advantage to be systematic[2]. Tetrys uniformly distributes the repair pack-

[1]This percentage does not include video exchanged through peer-to-peer (P2P) file sharing.

[2]An erasure code is said to be systematic if source packets

ets among the data packets and it recovers all lost packets within a small delay independent of the RTT. As a result, it has been shown in [13] that Tetrys significantly outperforms other erasure codes in the context of video transmission. Despite its performance, lost packets that are not recovered on time by Tetrys are considered as lost by the video decoder and these packets will impair the Quality of Experience (QoE).

In this paper, we propose "LD-Tetrys" (Late Decoding Tetrys), a solution based on on-the-fly erasure codes that counters a problem of late decoded packets being considered as lost. A video decoding scheme which is called late decoding takes the "late packets" (i.e., packets that arrive or are recovered after their playout deadline) into account in order to stop the error propagation. LD-Tetrys takes advantage of both Tetrys' properties: *i)* full reliability meaning that all lost packets are recovered as long as the redundancy ratio is higher than the average loss rate, *ii)* short decoding delay allowing to quickly use "late packets" to stop error propagation. We perform an analysis to illustrate why LD-Tetrys fits naturally late video decoding while the other schemes such as AL-FEC do not. We also develop an evaluation framework that is independent of video codec and network topology, thus allowing extensive evaluations. Experiment results show that LD-Tetrys has better performance in terms of video quality than normal decoding with Tetrys along the increase in loss rate, Group of Pictures (GOP) size. Furthermore, LD-Tetrys consistently outperforms both normal and late decoding with traditional block-based FEC codes.

The rest of the paper is organized as follows. Section 2 introduces briefly video encoding/decoding process and presents the difference between normal and late decoding with Tetrys. In Section 3, we tackle the distribution of the decoding delay and address why other approaches do not meet requirements for delay constrained applications. Section 4 presents our evaluation framework and simulation results. We discuss the related work to differentiate with our proposal in Section 5. Conclusion and future work are given in Section 6.

2. LATE DECODING WITH ON-THE-FLY ERASURE CODES

Most commonly used video codecs share the same encoding schemes. A GOP starts with an intra coded frame (I-Frame) followed by several predicted frames (P-Frames) or bidirectionally predicted frames (B-Frames). Subparts of a frame, denoted video coding unit (VCU)[3], are encoded separately. The P-frames of a GOP are coded as the difference between the VCUs of the frame itself and the VCUs of the I-Frames or the previous P-Frames' VCUs. B-frames are encoded as the difference of the VCUs from both previous and next P-frames[4]. The encoded VCUs are processed by a network abstraction layer (NAL) which packetizes and forwards them to the lower layer (e.g., AL-FEC or transport layer).

At the receiver side, packets are received and stored in a playout buffer, staying there until the playout delay expires and the video frames get decoded by the video decoder and played to the users. The purpose of the playout delay is to tolerate fluctuating network delay (also called jitter) or to provide enough time to retransmit or reconstruct the lost packets by an erasure code such as Tetrys. Unfortunately, this playout buffer cannot be arbitrary large. For instance, in the case of video-conferencing, it is preferable that the mouth-to-ear delay[5] remains below 100ms [15]. Delay higher than this value increasingly reduces the QoE perceived by the user. In many networks [10], the one way delay itself can be higher than 100ms, leaving room for a very small playout buffer if one wants to provide the user with the best possible QoE. The next two sections describe normal and late video decoding, how the bounded playout delay may hinder the performance of normal video decoding and why late video decoding might help.

2.1 Normal video decoding

At the playout time of a frame, a short playout delay leads to a high probability that some of the frame's packets are missing while that frame has to be decoded and displayed. Even if error concealment schemes aim at mitigating the impact of incomplete frame decoding, this impact still results in a highly degraded QoE. To worsen the picture, not only the incomplete frames will impact the video quality but also the subsequent P and B frames within a GOP due to error propagation. Fig. 1 describes the normal video decoding with the Tetrys erasure code. Indeed, as shown in the row denoted *Normal dec* if a given frame (say F_4) is encoded as the difference with its previous frame (F_3) which was incomplete at its playout time, the decoding of F_4 won't conform to the original image, in spite of all F_4's packets being received on time and all remaining packets of the GoP being available at the receiver. Even if there is no missing data in the subsequent frames (F_5), the errors will propagate until the end of the GOP. To prevent the missing packets from having such a strong impact, error concealment schemes are paired with an error resilience scheme on the encoding side. For instance, FMO allows VCUs to be allocated to several slice groups where each slice group is independent. Thus, FMO can mitigate the impact of packet loss at a cost of extra encoding bit rate.

2.2 Late video decoding

We believe that instead of using expensive error resilience schemes, video codecs should make the best use of the information available at the receiver side. As seen in the example of Fig. 1, when frames F_4 and F_5 are improperly decoded and displayed, Tetrys erasure code has already recovered missing packets P_3 and P_6, thus allowing the receiver to have all the required information to perform an impairment free decoding of these frames. The only condition to take advantage of the late arrival of the packets P_3 and P_6 would be to re-decode all or part of the VCUs of the previous errored frames (F_2 and F_3). This implies that at the playout time of F_4, all the packets up to P_8 must have been properly received. Frames F_2 and F_3 can be decoded without error to allow an impairment free decoding of F_4 and F_5, resulting

appear unaltered in the encoded output. As a result, some of the received packets are usable even if the decoding fails.

[3]Macroblock for H.264 and Coding Tree Unit for HEVC

[4]In the context of real time video transmission, the B-frame are usually ignored.

[5]In our case, the *mouth-to-ear* denotes the delay between the capture of the frame by a video source and its display at the receiver side.

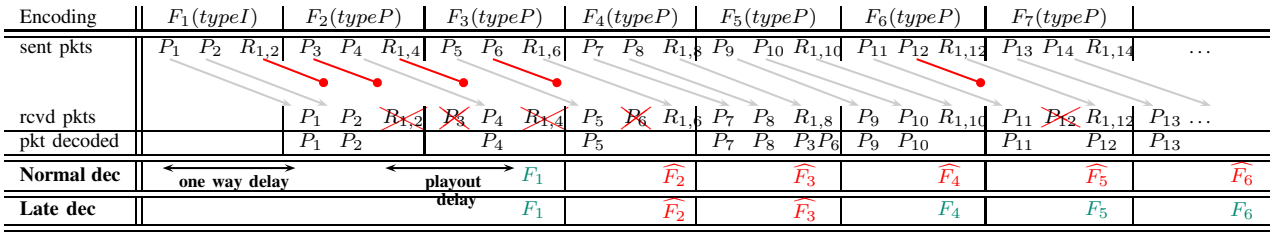

Figure 1: This figure presents the transmission using the Tetrys erasure code and video decoding (both normal and late) of one I-frame and 7 P-frames. For the sake of simplicity, we consider that each P-frame uses the previous frame as a reference, that each frame generates two packets and we represent the one way delay and the playout delay equal to the inter-frame delay. A frame F_i that contains error after its decoding is denoted $\widehat{F_i}$.

in the decoding pattern denoted *Late dec*. This decoding scheme is referred to as "late video decoding".

3. ERASURE CODES AND DISTRIBUTION OF THE DECODING DELAY

Among its various novel properties, in the context of delay constrained application, the most relevant contribution of Tetrys is the distribution of its packet decoding delay. Indeed, with Tetrys [13], the decoding process can be performed as soon as the number of redundancy packets received since the first unrecovered loss matches the number of missing source data packets.

3.1 Tetrys' encoding/decoding scheme

A coarse grained view of the encoding and decoding of the Tetrys erasure code can be observed in Fig. 1 which shows the packets generated by the video (ranging from P_1 to P_{12}) as well as the packets actually sent, received, and decoded by Tetrys. The source data packets are sent un-encoded while a redundancy packet denoted $R_{i,j}$ is a linear combination of all source data packets sent (indexed from i to j) but not acknowledged yet. In this example, for every two source data packets, Tetrys generates and sends one redundancy packet, leading to a redundancy ratio of $\frac{1}{3}$. We can see that packet $R_{1,2}$ is lost, but this loss has no impact on the video quality because P_1 and P_2 have been received. Then, packets P_3, P_6 and $R_{1,4}$ are also lost and at the reception of $R_{1,6}$, the receiver has two missing source data packets and only one repair packet ($R_{1,6}$). To recover these missing packets, the receiver has to wait until the reception of $R_{1,8}$. This implies when the number of missing source data packets (P_3 and P_6) matches the number of repair packets received since the first un-recovered loss ($R_{1,6}$ and $R_{1,8}$). For the late video decoding, it results that after the reception of $R_{1,8}$, the video frames F_2 and F_3 can be re-decoded without error and the user will perceive F_4 and F_5 error free. Further details such as Tetrys' acknowledgments scheme, complexity and decoding delay can be found in [13].

3.2 Why erasure block codes do not fit late decoding requirements

Consider a (n,k)=(3,2) erasure block code in Fig. 1, the loss of packet P_3 can not be recovered and late decoding is not be able to provide an error free decoding. Similarly, if we consider a (6,4) or a (9,6) code under the assumption that these codes start with P_1. To recover all lost packets in this

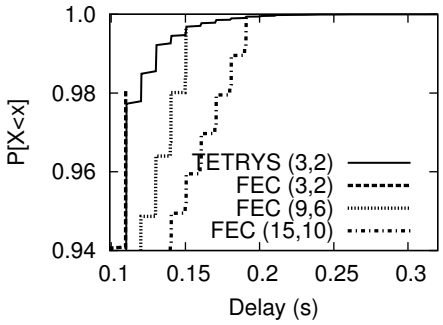

Figure 2: Distribution of Tetrys' and FEC's decoding delay (including a 100ms one way delay).

case, we need to increase the block size up to (12,8) where it would decode at the same time as Tetrys. Such large block code allows to recover more losses but it comes with a cost of a higher decoding delay. Indeed, the loss of packet P_{12} would be recovered by Tetrys with $R_{1,12}$ while the (12,8) block code would need to wait until the end of the block (after the reception of P_{16}) and hence the video decoder wouldn't be able to decode F_6 and F_7 without errors.

This toy example suggests that if one wants to use an erasure block code with late-decoding, it should be configured with large block size to reduce the residual loss that impairs the video decoding. This induces a higher decoding delay coming with the cost of a higher playout delay which impacts the QoE. On the other hand, Tetrys allows recovering all lost packets and ensures that they'll be recovered with the smallest possible delay, thus reducing the impact of error propagation as much as possible. In order to have a clear idea on the advantage of Tetrys' decoding delay, Fig. 2 shows the distribution of the decoding delay of Tetrys and FEC under the assumption that packets are sent by a video codec generating 30 frames per second with GOP size of 30 images, using only I and P frames with 15 packets for an I-frame and 3 packets for a P-Frame. For the sake of readability, the packet rate is smoothed to a CBR of 102 packet/s. Fig. 2 clearly shows the tradeoff between the block size and the decoding delay. We can see that for each FEC block size, FEC has a different maximum decoding probability which is achieved for a given value of the delay (e.g., 0.15 sec for

FEC(9,6)). For every delay smaller than this one, Tetrys recovers more packets than FEC. For FEC, waiting longer than this delay doesn't allow to decode any other packets while Tetrys will recover all lost packets.

The main result is that with normal decoding, Tetrys is better than or equal to FEC while with late decoding with Tetrys always provides improvements compared to FEC.

3.3 ARQ schemes do not help

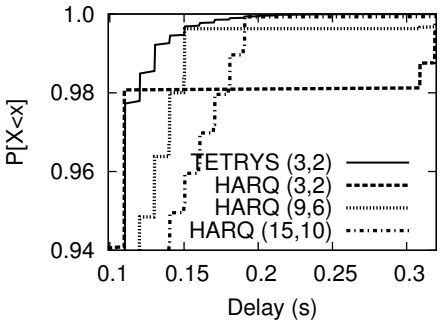

Figure 3: Distribution of Tetrys' and HARQ's decoding delay (including a 100ms one-way-delay).

In order to recover all lost packets without the hindrance of large block size's decoding delay, hybrid ARQ (Automatic Repeat-reQuest) schemes known as HARQ combine small erasure codes with the retransmission of the lost packets that are not recovered by the erasure codes. If a lost packet can not be recovered by the erasure codes, the packet recovery is delayed by at least one RTT as its retransmission will be asked by the HARQ scheme. As a consequence, it can bring an improvement only if the RTT is such that the packet recovery will be quick enough to minimize the number of frames that'll be shown to the user with an error (i.e., not more than a few multiple of 33ms in case of the 30 fps video). Fig. 3 shows the distribution of the recovery delay for HARQ and Tetrys under the same setup as in Fig. 2 with a RTT of $200ms$[6]. As expected, we can see that the distribution of the decoding delay by the erasure code is the same as on Fig. 2 whereas the packets requiring retransmission will arrive one RTT later (after 0.3s) which means that errors will spread during at least 9 frames before the decoder will get a chance to remove them.

The main consequence is that except for small RTT or high value of the loss rate [13], Tetrys' decoding delay is always much smaller than the one of HARQ and hence, we can expect that late decoding will provide much better performance with Tetrys. As a result, in the evaluation, we only show the comparison of Tetrys and FEC for late decoding.

4. EVALUATION OF LD-TETRYS

Our evaluation toolchain is summarized in Fig. 4. The JM [1][7] H.264 reference software encodes an YUV source file with

[6]A RTT of 200ms is roughly similar to the one experienced between US and Europe [10].

[7]Note that our framework can also process any video stream generated by GPAC [7], including HEVC.

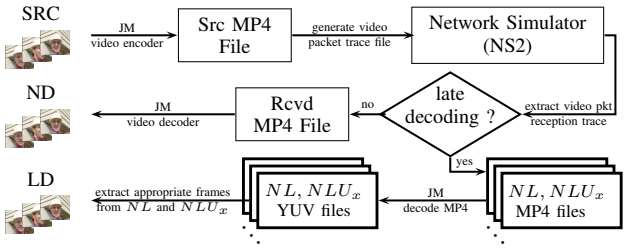

Figure 4: Testbed for normal decoding and late decoding evaluation using the JM software, the network simulator NS2 and off-line late decoding. "SRC", "ND" and "LD" respectively stands for the YUV file obtained from the source, from normal decoding and from late decoding

the appropriate parameters and generates a trace of RTP packets. The transmission of the RTP packets on the network is evaluated using the `ns-2` network simulator [8] in which we implemented an end-to-end version of Tetrys and application layer erasure block codes (e.g., AL-FEC).

Normal video decoding is obtained by simply feeding the JM decoder with the RTP trace from which we remove the the packets received after their playout time. Late decoding could be obtained through minor modification of the JM video decoder software but it would come with the downside of being dependent to a given software and codec. We overcome this by using the output of normal video decoding feeded with various RTP traces: NL and NLU_x respectively referring to the case where they contains No Loss and No Loss Until the x-th packet. At the playout time of a frame F_i, if every packet up to F_i has been received or recovered, the late decoded version of F_i is chosen from the output of the NL trace. Otherwise it is chosen from NLU_x with x, the oldest missing packet at the playout time of F_i. Following the example of Fig. 1, frame F_1, F_4 and F_6 are picked up from NL while frames F_2 and F_3 are picked up from NLU_3. This framework allows to evaluate the combination of Late Decoding and erasure codes in various contexts such as studying the impact of the network topology, the cross traffic interaction with other applications as well as the impact of the transport protocol and congestion control.

4.1 Evaluation parameters

Our evaluation assumes a one-way-delay of 100ms and the delay of the return path has no impact on our evaluation. We define an experiment as the evaluation of a given coding scheme (e.g., ND-Tetrys, FEC, LD-Tetrys), with a value set for each parameter: GOP size, packet loss rate (PLR) and playout delay. An experiment is performed for each of the 6 reference videos in CIF format that last 300 frames each {akiyo, foreman, news, bus, highway, football}. For each reference video, the results are averaged over fifteen runs. The average PSNR corresponding to an experiment is the average of 90 simulations (15 runs $\times$ 6 reference videos). By default, the value of k is set to 6 ($R \approx 14\%$), the video is encoded with a maximum of 3 reference frames, without FMO, a GOP size of 30 frames with baseline profile having only I and P frames at a rate of 30 frames per second.

4.2 LD with various parameters

This section describes a subset of our experiments and shows the gain brought by LD-Tetrys compared to ND-Tetrys and LD-FEC as a function of PLR, GoP size and playout buffer size. We observe similar improvement with other parameters such as number of reference frames, video profile or intra macro block refreshing.

4.2.1 LD (PSNR) as a function of loss rate

Fig. 5 shows the PSNR as a function of the loss rate for LD-Tetrys, ND-Tetrys and LD-FEC with various block sizes. The redundancy ratio is set to 14% for every schemes. ND-Tetrys and LD-Tetrys achieve similar performance for PLR values ranging from 0 to 2% while with higher PLR values, LD-Tetrys improves the results by up to 4dB. For a 10%

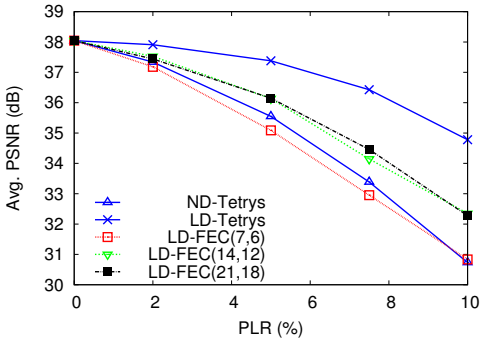

Figure 5: PSNR as a function of PLR with R=14%.

PLR, LD-FEC improves the PSNR by up to 1.5dB compared to ND-Tetrys but LD-Tetrys consistently outperforms LD-FEC with a maximum of 2.5dB in this context[8]. Another downside of LD-FEC is that it is not always the same block size that provides the best performance. The configuration of FEC with late decoding would thus be harder compared to LD-Tetrys which only require to specify the redundancy ratio. This result corroborates the insight brought in Section 3 arguing that the distribution of Tetrys' decoding delay best fits the requirement of late video decoding. Due to the lack of space and significance, the remaining of this paper won't show any FEC based results.

4.2.2 LD (PSNR) as a function of GOP size

The goal of this experiment is to study if the improvements brought by LD-Tetrys hold when the GOP size varies. Fig. 6 shows the PSNR of ND-Tetrys and LD-Tetrys as a function of the GOP size for packet loss rate of 2%, 5% and 10%. As expected, the larger the GOP size leads to the lower the average PSNR of ND-Tetrys. This is because the error propagation is spread until the end of the GOP before they get cleared by the next I-Frame. Surprisingly, the figure shows that the GOP size has no impact on LD-Tetrys. This can actually be explained by fact that error propagation is limited by Tetrys' decoding delay which means that all the errors are removed after a decoding event. For appropriate values of the redundancy ratio, the decoding delay is much smaller

[8]Note that with a higher number of reference frames and a larger GOP size, we noticed that LD-Tetrys allows gains up to 10dB compared to LD-FEC.

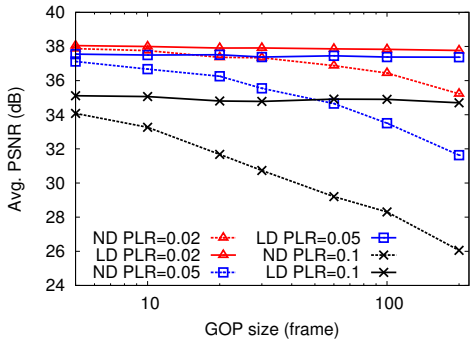

Figure 6: PSNR of LD-Tetrys and ND-Tetrys as a function of GOP size.

than the GOP size, thus significantly saving the encoding bit rate by adding new I-Frames.

It is interesting to note that LD-Tetrys allows the safe use of large GOP size. For instance, the throughput of the Akiyo reference video is 45% higher with GOP size of 30 frames compared to infinite GOP. LD-Tetrys paves the way toward a significant improvement in the QoE throughput tradeoff.

4.2.3 LD (PSNR) and interactivity

The playout delay is a key parameter for the QoE and it should be kept as small as possible. However, a short playout delay allows less time for packet recovery. This section aims at demonstrating that LD-Tetrys isn't significantly affected by the variations of the playout delay. Fig. 7 shows the PSNR as a function of the playout buffer for values ranging from 60ms to 400ms. Results are shown for LD-Tetrys and ND-Tetrys with PLR values of 2%, 5% and 10%. We

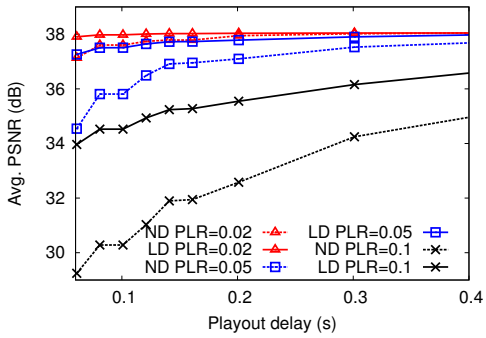

Figure 7: PSNR of LD-Tetrys and ND-Tetrys as a function of playout delay.

can see that for any playout delay, LD-Tetrys outperforms ND-Tetrys. When decreasing the playout delay from 400ms to 60ms, the PSNR drop of LD-Tetrys remains limited to respectively 0.1dB, 0.8dB and 2.5dB for PLR of 2%, 5% and 10% while the PSNR of ND-Tetrys is dropped by 1dB, 3.2dB and 6dB, respectively.

It is also interesting to compare the slope of the curves. For normal decoding, the slope is higher for small values of the

playout delay. It means that the smaller the playout delay the stronger will be the impact of a playout delay reduction. With LD-Tetrys, the slope of the curve is slight and almost steady. This means that even without playout buffer, LD-Tetrys still brings an improvement compared to unprotected video. This is due to the fact that even if Tetrys won't be able to recover the frames' packets before their playout time, the error induced by the packet will still be limited by Tetrys' decoding delay.

This property is a great advantage in case of high or varying network conditions. It may also ease the use of bi-directionally predicted frames (B-frames) in the context of live encoding for delay constrained applications. Indeed, if we assume a frame rate of 30 fps with an IBPBP pattern, the encoding of a B-frame must wait for the next frame to be captured an encoded. B-frames will be available for network transmission 33ms after their capture while with an IPPPP frame pattern the only delay is due to computation time. When decoding, the frame period must remain constant, delaying the entire video by 33ms. It reduces the amount of time available for error recovery if one wants to keep the same *mouth-to-ear* delay than with an IPPPP pattern. Fig. 7 shows that a reducing the playout delay from 100ms to 66ms impacts LD-Tetrys by a decrease of only 0.1dB (resp. 0.3 and 0.5dB) when the PLR is 2% (resp. 5% and 10%) while saving a significant amount of throughput thanks to the B-frames (e.g., 16 et 20% for the akiyo and football reference video with our default parameters).

5. RELATED WORK

This decoding scheme, often denoted as "late decoding", has been studied in [5, 9] and [16]. In [5], the author proposed a cell buffering [5] in the context of ATM networks. During congestion, instead of being dropped, video cells are buffered in switches and sent back to the network thus generating late packets that are useful for "late decoding". In [9], the authors suggest the use of "late decoding" to cope with packet loss in the Internet by coupling the use of FEC codes and the retransmission of lost packets that are not recovered by these codes. More recently, the authors in [16] also proposed to use "late decoding" in combination with erasure block codes. Despite those works have demonstrated that "late decoding" brings an advantage compared to normal decoding, this scheme has never received much attention. It is possibly because they still require the use of error resilience schemes to temper the fact that erasure block codes have a non negligible residual loss rate or, in the case of retransmission, the fact that errors are perceptible for a duration as big as the retransmission delay which may take a significant time. If late decoding is combined with an error recovery scheme that allows full reliability within a short delay, all previous improper decoded frames can be quickly corrected to stop the error propagation, leading to a good QoE without requiring any error resilience scheme and thus saving a significant amount of bandwidth. Such error recovery schemes (e.g., Tetrys) have been recently proposed, thus allowing to leverage the late decoding's benefits.

6. CONCLUSIONS

In this paper, we proposed a late decoding solution based on Tetrys (called LD-Tetrys) to deal with delay constrained applications. Our analysis showed that LD-Tetrys fits well to the requirements for late video decoding while the other schemes (e.g., FEC, HARQ) do not. We also developed an evaluation framework which is independent of video codec and network topology. Simulation results acknowledge that LD-Tetrys' performance is better than the normal decoding with Tetrys, the original Tetrys. Furthermore, LD-Tetrys consistently outperforms the traditional block based erasure codes such as AL-FEC in terms of video quality. For future work, we are working on the theoretical modeling and analysis. We also expect to perform extensive experiments to obtain a complete evaluation.

7. ACKNOWLEDGMENTS

The authors would like to thank Prof. Paul Amer for valuable comments about this work.

8. REFERENCES

[1] H.264/AVC JM Reference Software, http://iphome.hhi.de/suehring/tml/.

[2] Cisco Visual Networking Index: Forecast and Methodology, 2012-2017.

[3] RMCAT IETF working group http://datatracker.ietf.org/wg/rmcat/.

[4] H. Alvestrand. Overview: Real Time Protocols for Brower-based Applications. Technical report, IETF, 2013.

[5] M. Ghanbari. Postprocessing of late cells for packet video. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(6):669–678, 1996.

[6] J. Lacan and E. Lochin. Rethinking reliability for long-delay networks. In *Proceedings of International Workshop on Satellite and Space Communications IWSSC*, Oct. 2008.

[7] J. Le Feuvre, C. Concolato, J.-C. Dufourd, R. Bouqueau, and J.-C. Moissinac. Experimenting with Multimedia Advances Using GPAC. In *Proceedings of the 19th ACM International Conference on Multimedia*, pages 715–718, NY, USA, 2011.

[8] S. B. Lee et al. Improving simulation for network research. Technical report, University of Southern California, 1999.

[9] I. Rhee and S. Joshi. Error recovery for interactive video transmission over the Internet. *IEEE Journal on Selected Areas in Communications*, 18(6):1033–1049, 2000.

[10] S. Shakkottai, N. Brownlee, A. Broido, and k. claffy. The RTT distribution of TCP flows on the Internet and its impact on TCP based flow control. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), Mar 2004.

[11] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012.

[12] J. K. Sundararajan, D. Shah, and M. Médard. ARQ for network coding. *IEEE International Symposium on Information Theory*, pages 1651–1655, July 2008.

[13] P.-U. Tournoux, E. Lochin, J. Lacan, A. Bouabdallah, and V. Roca. On-the-fly coding for time-constrained applications. *IEEE Trans. on Multimedia*, 13, 2011.

[14] Y. Wang and Q.-F. Zhu. Error control and

concealment for video communication: a review. *Proceedings of the IEEE*, 86(5):974–997, 1998.

[15] S. Wenger. H.264/AVC over IP. *IEEE Trans. on Circuits and Systems for Video Technology*, 13:645–656, 2003.

[16] J. Xiao, T. Tillo, C. Lin, Y. Zhang, and Y. Zhao. A real-time error resilient video streaming scheme exploiting the late- and early-arrival packets. *IEEE Trans. on Broadcasting*, (99):1–1, 2013.