EFFICIENT LIGHTWEIGHT VIDEO PACKET FILTERING FOR LARGE-SCALE VIDEO DATA DELIVERY

Xavier Corbillon[⋆] *Florian Boyrivent*[⋆] *Grégoire Asselin De Williencourt*[⋆]
Gwendal Simon[⋆] *Géraldine Texier*[⋆] *Jacob Chakareski*[†]
[⋆] Télécom Bretagne / IRISA, France [†] University of Alabama, USA

ABSTRACT

When the video bit-rate is greater than the available network bandwidth, the video stream suffers from packet loss due to packets that need to be dropped. A streaming server (or a network proxy) can implement a proactive packet filtering strategy, which is to voluntarily block (not forward) some packets in the event of such bandwidth mismatch. The challenge is to decide which packets to block so that the quality of the video at the client side is maximized with regards to the available bandwidth. Previous proposals aimed to use meta-information from the video encoding or to pre-process the multimedia data. Our goal is to design a lightweight strategy, which only uses video metadata available from the video file container. We demonstrate on a set of HEVC videos that out lightweight packet filtering algorithm performs as well as more complex strategies. Moreover, the video quality remains high despite a large number of blocked packets, while a random selection of dropped packet leads to a significant quality drop.

1. INTRODUCTION

Content delivery providers have to deal with situations where the average bit-rate of the video is greater than the available capacity of the network link. In current adaptive streaming technologies, such mismatch between the bit-rate and capacity happens for the duration of a video segment, which ranges from 2 seconds (s) to 10 s. It can be due to an unexpected drop of bandwidth, the failure of a too optimistic rate-adaptive strategy, an increase of the video bit-rate for this video segment, or a combination of these causes. If nothing is done, this bandwidth shortage results in drop of packets at the bottleneck of the link regardless of the packet content and without any control from the endpoints.

To conform the video bit-rate to the available bandwidth, the streaming server can implement a proactive *packet filtering* strategy, which is the process of blocking some data packets of the video stream at the server side. Video packet filtering aims at minimizing the impact of packet loss on the Quality of Experience (QoE) at the client side. It blocks the packets that carry data from video frames such that the loss of these frames minimizes the video distortion. Packet filtering strategies based on the video structure lead to improvement in the perceived QoE of the user [3].

The video packet filtering solutions proposed so far consider either to pre-compute information on the video or to decode the video to extract deep coding information [1, 9, 12, 15]. It is challenging to implement such techniques for the popular content providers, which manage a catalog of several millions videos and thousands of new video requests per second.

The work of Jacob Chakareski was partially supported by the NSF award CCF-1528030

We propose *lightweight* packet filtering strategies, which do not require pre-processing, nor analysis of the encoded multimedia content. The only inputs that we consider are the video metadata (frame size, frame type, and frame dependencies) that can be extracted from standard multimedia containers. We focus on the most recent video encoding trends, including high-resolutions (1080p) videos encoded with High Efficiency Video Coding (HEVC), and the encoding settings that are now used in interactive multimedia applications such as high frame-rate and customized Group of Picture (GOP).

Our contribution is threefold. First, we formally introduce the packet filtering optimization problem, and we propose an algorithm to compute the optimal solution based on a well-known distortion model [2]. Second, we introduce our practical, lightweight algorithms based on video metadata. We study the combination of metadata that enables the identification of the least impacting frames regarding QoE. We evaluate the Multiscale - Structural Similarity (MS-SSIM) video quality metric [18] that can be achieved when our packet filtering algorithm is implemented on a set of videos for various bandwidth shortage conditions. Third, we analyze the behavior of our packet filtering when the video streams feature some emerging encoding formats: the open-GOP concept, a frame-rate of up to 60 fps, and video encoding without bi-directionally-encoded (B) frames.

The paper opens two new perspectives. From a practical perspective, we show that efficient packet filtering solutions can be implemented with minimum computational requirements. In particular, our lightweight algorithm preserves a high QoE despite significant bandwidth shortage, for example the average MS-SSIM is higher than 0.95 when 10 % of the packets must be blocked. The integration of packet filtering into new content delivery techniques becomes an option that should be considered in the future. From a theoretical perspective, the results call for new developments on the distortion estimation, especially with regards to the new video encoding settings.

2. STATE OF THE ART

The relation between the average subjective QoE and the packet loss has been a research topic for years [19]. Predicting the video quality can be based on the level of motion and the scene complexity [5, 10] or using information on mismatched blocks [14]. The researchers focus on estimating the overall quality of the video. However, they do not indicate which frames are best to block to meet a bandwidth constraint while minimizing the impact on the video quality. Another related research topic is about no-reference video quality estimation models in case of frame losses, typically the Commulative Mean Square Error (CMSE) or the Peak Signal Noise to Ratio (PSNR) [2, 15]. Paluri et al. [15] typically estimate CMSE for the loss of a unique frame using deep coding information

extracted during the encoding phase. Those techniques need metadata available at the encoding time or high computational resources for the computation.

By using machine learning approaches, Staelens et al. [17] extract useful information from the compressed video to estimate the perceived quality. The main goal of those studies is to be able to estimate the visibility of packet loss at the client to get the average perceived QoE. But they do not study frame level packet loss.

Mehdian and Liang [12] studied a scheduler to send a subset of frames to respect bandwidth constraint while maximizing the overall video quality. They characterize the properties of their optimal transmission scheduler in terms of frame dependency and they demonstrate that their exists a canonical form for the solution. They also proposed a quadratic algorithm to solve the optimal scheduling problem. To evaluate the overall quality, they consider that a frame cannot be decoded if any dependency is missing while we stick to the real conditions in which the decoder can partially decode a frame even if some of its dependencies are missing.

Adaptive bit-rate streaming technologies such as the Dynamic Adaptive Streaming over HTTP (DASH) standard aim at coping with variable network conditions and maintaining the best QoE during the playback. The server offers several video *representations*, which differ by their spatial resolution and their average bit-rate. Each representation is cut into segments, whose duration ranges from 2 s to 10 s. The client must select one video representation for each segment based on estimations of network conditions in the next seconds. Once the client selects a given video segment, this selection is irreversible, so the server streams this video segment until the time to select the next segment. However, as shown by Huang et al. [6], the segment selection is a hard task, which is often inaccurate in commercial solutions. Despite research efforts [16], mismatch selections still happen resulting in bandwidth shortage. To the best of our knowledge, no previous work related to adaptive streaming has dealt with adaptation *during* the segment download. Our proposal addresses this gap.

3. PROBLEM DESCRIPTION

We denote by $\mathcal{P}$ the set of data packets for the whole video, each packet being the size of the Maximum Transmission Unit (MTU) of the link. We suppose the streaming server has a way to estimate the available bandwidth on the link (for example by probing the link [13] or by using network parameters [11]). According to this prediction, the total number of packets that can be delivered on time is less than $|\mathcal{P}|$. Let N be this difference, i.e. N is the minimum number of packets in $\mathcal{P}$ that must be blocked to fit with the bandwidth constraints. We aim at selecting the set of packets to block so that the distortion is minimized. To achieve this objective, we take into account the frame structure of the video.

We work with the HEVC/H.265 codec. The HEVC encoding, like Advanced Video Coding (AVC)/H.264, uses a hierarchical structure to exploit temporal and spatial redundancies in the video to efficiently compress it. Pictures from the original video are encoded into three types of frames: intra-predicted (I) frames, inter-predicted (P) frames and bidirectional (B) frames. The encoder splits each pictures into different slices, which contains different lines of the encoded picture, and can also split them into different tiles (rectangular area of the picture that can be decoded independently from each other).

Let $\mathcal{F}$ be the set of frames for the whole video. Each frame is carried out by a subset of packets. We denote by n_j the number of packets that carry out the data related to frame $j \in \mathcal{F}$. The management of a packet loss (also known as error concealment) depends on the strategies the decoder implements and on the information contained in the packet. The decoder is sometimes able to extrapolate the missing information, but most frequently it has to drop the whole frame because key information are missing. To stick to the universality of this study, we consider that any packet loss in an encoded frame leads to the whole frame not being decodable, which results in a distortion.

Let $D(\mathcal{F}')$ be the distortion due to the loss of the set of frames $\mathcal{F}' \subseteq \mathcal{F}$. It is important to recall that the distortion function is not an additive function of the frames: $D(\{j_1, j_2\})$ is not always equal to $D(\{j_1\}) + D(\{j_2\})$. Indeed, Liang et al. [8] have demonstrated that the distortion generated by the loss of frames close to each other is greater than the sum of the distortion generated by the isolated loss of the same frames. The linear approximation is valid only when lost frames are far enough from each other (typically in different GOPs).

Problem Formulation. We aim at deciding the set of packets $\mathcal{P}' \subseteq \mathcal{P}$ such that $|\mathcal{P}'| \geq N$ and the distortion generated by the frame losses due to the non-delivery of packets in $\mathcal{P}'$ is minimum. Since the loss of only one packet in a frame results in the whole frame being decodable, it is obvious that the decision of blocking a packet in a frame j means that all the packets related to frame j are blocked too. Therefore, the problem can be re-formulated at a frame level: we aim at deciding the set of *frames* $\mathcal{F}' \subseteq \mathcal{F}$ such that the distortion due to the loss of $\mathcal{F}'$ is minimum and the sum of all packets related to the frames in $\mathcal{F}'$ is greater or equal to N. Let x_i be a binary variable such that:

$$x_j = \begin{cases} 1, & \text{if frame } j \text{ is blocked} \\ 0, & \text{otherwise} \end{cases}, \forall j \in \mathcal{F}$$

The problem is formally defined as:

$$\underset{\mathcal{F}' \in 2^{\mathcal{F}}}{\textbf{minimize}} \quad D(\mathcal{F}') \tag{1a}$$

$$\textbf{subject to} \quad j \in \mathcal{F}' \text{ if } x_j = 1 \tag{1b}$$

$$\sum_{j \in \mathcal{F}'} x_j \times n_j \geq N \tag{1c}$$

Problem Analysis. The computation of the optimal solution requires pre-computing all the possible distortion values. This pre-computation increases exponentially with the number of frames in $\mathcal{F}$ since there exist $2^{|\mathcal{F}|}$ possible subsets of frame losses. This problem is a variant of the 0-1 knapsack problem, which is NP-complete [7]. The objects are the frames and the sack corresponds to the available bandwidth. The weight of each frame $j \in \mathcal{F}$ is n_j (the number of packets related to j) and the value of each frame j is the distortion due to the loss of j. This problem is a variant of the original knapsack problem because the value of each frame (here the distortion) depends on the other lost frames. Note that the 0-1 knapsack problem is a special case of this variant where the distortion function is the sum of the distortions of the lost frames, so it is easy to prove that this problem is also NP-complete.

4. SOLUTION WITH DISTORTION CHAIN MODEL

4.1. Distortion Chain Model

Solving the optimal problem introduced in Section 3 requires the full knowledge of the function D, *i.e.* the pre-computation of all $2^{|\mathcal{F}|}$ possible distortion values. In practice, the exponential time complexity prevents the use of the real D function. To have a

reference to which we can compare our lightweight algorithm, we use the Distortion Chain model first introduced by Chakareski et al. [2] for computing an estimated function $\widetilde{D}$ of D. We use the order 1 Distortion Chain model ($DC1$). We denote by $\widetilde{D}(\mathcal{F}')$ the estimated distortion that is generated by the loss of the set of frames $\mathcal{F}' \subseteq \mathcal{F}$ and by $D(j \mid j')$ the additional distortion generated by the loss of the frame j knowing that frame j' was lost: $D(j \mid j') = D(\{j', j\}) - D(j')$ with $j' < j$ and $(j', j) \in \mathcal{F}^2$. Then, the DC^1 distortion can be expressed as follow:

$$\widetilde{D}(\{j_{n_1}, j_{n_2}, \ldots, j_{n_N}\}) = D(j_{n_1}) + \sum_{i=2}^{N} D(j_{n_i} | j_{n_{i-1}}) \quad (2)$$

Please note that if $|\mathcal{F}'| \leqslant 2, \widetilde{D}(\mathcal{F}') = D(\mathcal{F}')$.

When the distortion estimation model is used instead of computing the distortion for all subsets, the accuracy of the model is lower, especially if the packet loss rate is greater than 8%, but the number of distortion computation is significantly lower: $O(|\mathcal{F}'|^2)$ against $O(2^{|\mathcal{F}'|})$.

4.2. A Dynamic programming solution

In Section 4.1, we introduced the model used to estimate the distortion generated by any set of frame loss in the video. We now discuss the technique we use to solve the problem of finding the optimal set of frames to block, subject to the minimum number of packets that have to be dropped.

When the weight of the objects (here the number of packet needed to transport the frames) is an integer, the knapsack problem can be solved using *dynamic programming* with a pseudo polynomial time and memory complexity. We cannot directly use classical dynamic programming algorithm to solve our problem because it supposes the value of each object to be a constant. In the case of the $DC1$ model, the distortion depends on which other frame is blocked.

Dynamic programming can be used if the optimal solution of our problem can be computed from the combination of optimal solutions of sub-problems. We define $m[i, P]$ to be the minimum distortion that can be generated if the last frame lost in $\mathcal{F}$ is the frame with id i such that at most P packets are sent. Then the minimal distortion for the whole problem is :

$$\min_{i=0}^{|\mathcal{F}|}(m[i, |\mathcal{P}| - N - \sum_{j=i+1}^{|\mathcal{F}|} n_j]) \quad (3)$$

Each $m[i, P]$ can be recursively computed as follow:

$$m[0, P] = \begin{cases} 0, & \text{if } P \geqslant 0 \\ +\infty, & \text{otherwise} \end{cases} \quad (4a)$$

$$m[i, P] = \min_{k=0}^{i-1}(m[k, P - \sum_{j=k+1}^{i-1} n_j] + D(i \mid k)) \quad (4b)$$

With $D(i \mid 0) = D(i)$.

The time complexity of this dynamic programming algorithm is $O(|\mathcal{F}|^2|\mathcal{P}|)$ and the memory complexity is $O(|\mathcal{F}||\mathcal{P}|)$. To know which frame is blocked in the minimum distortion solution, we only need to store in the vector m the list of frames we block at each step.

Name	Frame Type T_j	Dependencies D_j	Frame Size S_j^+	Frame Size S_j^-	Random R
Random					+++++
Type	++++				+
Dependencies		++++			+
DropSmall			+++++		
DepDropSmall		++++	+		
DepDropBig		++++		+	
HybridDropBig	++	+++		+	

Table 1. Algorithms parameters

5. OUR LIGHTWEIGHT ALGORITHMS

5.1. Evaluation Function

An efficient packet filtering relies on an evaluation function that singles the blocking frames out according to their importance. The more important a frame is, the higher this impact on the QoE will be. We identified three relevant indicators to estimate the importance of a frame $j \in \mathcal{F}$, namely the frame type (T_j), the number of dependent frames (D_j), and the frame size (S_j^+). Those three indicators are transformed to real numbers normalized between 0 and 1.

Frame Type Selecting the frames to block according to their frame types favors the blocking of B frames rather than P frames or I frames. An I frame contains by itself the full description of an image and so can be decoded independently of any other frames. Moreover, it is generally the start of a dependency chain in the video, so a missing I frame has a strong impact on the QoE because multiple other frames have direct or indirect references to it.

Dependency Blocking a frame that is needed by a large number of other frames increases the QoE degradation since errors are propagated to all the dependent frames. To avoid complex pre-computation, we restrict the frame dependency metric to the first order dependency, that is, a frame dependency corresponds to the number of frames that directly need the current frame to be decoded.

Frame Size The size of the frame gives an indication on the quantity of information provided. Moreover two frames have rarely the same size. Hence the frame size can help to decide which frame can be blocked when several are eligible. However, it is difficult to establish if bigger frames should have a high or a small evaluation value. Given the minimum number of packets N that must be blocked, we cannot tell *a priori* if it is better to lose many small frames or a few big frames. Thus, we introduce two versions of the frame size evaluation: S_j^+ where bigger frames have a higher evaluation value compare to small frames and $S_j^- = 1 - S_j^+$ where smaller frames have a higher evaluation value compare to big frames.

Finally we introduce a random number R_j uniformly taken between 0 and 1 to decide which frame to block between equivalent candidates.

The evaluation function is defined as a multi-criteria linear function with the constants $\tau, \delta, \sigma^+, \sigma^-$, and ρ, and is denoted as:

$$E_{\{\tau,\delta,\sigma^+,\sigma^-,\rho\}}(j) = \tau T_j + \delta D_j + \sigma^+ S_j^+ + \sigma^- S_j^- + \rho R_j, \quad \forall j \in \mathcal{F}$$

The evaluation function is defined at the frame level but the blocking decision is performed at the packet level. Therefore, the evaluation value for a packet is the evaluation value of the transported frame. Recall that any missing packet in a frame leads to the complete loss of the frame (it is equivalent to lose every packet of the frame).

5.2. Algorithms parameters

We now introduce the eight evaluation function parameter sets we selected to emphasize the influence on the QoE of the different indicators introduced in Section 5.1. Those evaluation function are summarized in Table 1.

First we introduce the random evaluation function *Random*, defined as $E_{\{0,0,0,0,5\}}$, that sets a random evaluation value for each frame. This function simulates the behavior of random packet loss in the video, which is the typical behavior at a bottleneck link.

We define the function *DropSmall* as $E_{\{0,0,5,0,0\}}$, *Type* as $E_{\{4,0,0,0,1\}}$, and *Dependencies* as $E_{\{0,4,0,0,1\}}$ to emphasize the influence of a single indicator, respectively the frame size, the frame type, and the frame dependencies. The *HybridDropBig* function, defined as $E_{\{2,3,0,1,0\}}$, is based on our intuition that a good evaluation function should consider the frame type and the frame dependencies. The first discriminating indicator is the number of dependencies. In case of a draw, we look at the type of the frame and then we prefer to lose big frames. The functions *DepDropSmall*, defined as $E_{\{0,4,1,0,0\}}$, and *DepDropBig*, defined as $E_{\{0,4,0,1,0\}}$ focus on the way to prioritize frames according to the packet size in case of a draw in the frame dependencies metric.

6. OUR TESTBED

We now describe the testbed that we set up to evaluate the behavior of our lightweight packet filtering proposals. To improve the universality of our study, we work with eleven raw YUV videos of 10 s, which are representative of the diversity in multimedia videos, including crowd, water, explosion, and sport. Our testbed emulates the streaming of HEVC videos.

6.1. Video transmission chain

Our testbed is represented in Figure 1. It is divided into six main components. First the encoder compresses the original YUV video into an HEVC video using the *ffmpeg* encoder. Then, we recover the metadata from the HEVC video, i.e. the number of frames in the video, the frames coding identifier, their types (I, P, B), their sizes, and their numbers of dependencies. Using such metadata, we then apply the chosen blocking strategy to select which packets to block in order to meet the bandwidth shortage constraint. We then remove from the original HEVC video any encoded frame that contains a blocked packet. This HEVC video with missing frames is given to the *libav* / *libx*265 decoder in order to generate a YUV representation of the decoded video (as a client would have decoded it). When the decoder is not able to generate a picture because too much information is missing, an error concealment strategy is used: we repeat the last displayed picture.

Finally, we use a full reference objective metric named MS-SSIM to compare the YUV video generated by the emulation chain with the original YUV. The MS-SSIM is computed for each frame with the Video Quality Measurement Tool (VQMT) [4] and we keep only the average result.

6.2. Encoding parameters

In our first set of evaluations, we use four videos encoded with HEVC at a spatial resolution of 1920×1080 pixels (p), an average bit rate of 20 Mbps, and a temporal frame rate of 25 frames per second (fps). We set a static pyramid GOP structure of 32 frames $IPBBB$. The distance between two consecutive anchor frames (I or P) is four frames; the distance between two I frames is 32 frames.

In our second set of evaluations we study the impact of different GOP structures on our blocking strategies. For that we used the same four original YUV used to encode the video from the first set of evaluation except that here we encoded those video using different HEVC parameters. We used three different GOP structures. First, we use *libx*265 dynamic scene cut parameter to allow an adaptive I frame placement in the video and we keep the frame rate at 25 fps. Second, we keep the static GOP structure from the previous set of evaluation but we switch to a frame rate of 60 fps. Finally, we test the HEVC no delay profile by having no encoding B frames. We keep one I frame every 32 frames.

7. EXPERIMENTAL RESULTS

7.1. Impact of Indicators

We first compare the different blocking strategies per indicator (frame type, size, and dependency). First we evaluate the impact of the different indicators individually to identify the one that gets the best average video quality; then we combine this best indicator with others to study their impact; finally we compare the best combination of indicators with the Distortion Chain model results. Each sub-figure in Figure 2 displays the average MS-SSIM for the four representative videos.

We first show that any of the three indicators individually (*Type*, *Dependencies* or *DropSmall*) enables deciding the frames to block such that the quality of the received video is significantly better than random dropping. The average MS-SSIM for *Type*, *Dependencies* and *DropSmall* on Figure 2a are always over the average MS-SSIM of random packet loss. The indicator that provides the best result is the number of dependencies: *Dependencies*. It is an intuitive result because the more dependencies a frame has, the more frames are impacted if it is dropped. A packet filtering strategy only based on dependencies can block up to 17 % of the video packets but still have the same QoE as the video with 2 % of randomly chosen packet losses.

We now aim at selecting the second best indicator to use (after the frame dependency) instead of using random to address frame ties. We observe in Figure 2b that, when two frames have the same number of dependencies, blocking the biggest frames is better than blocking the smallest frames. This is counter-intuitive, since the size of a frame represents the amount of key multimedia information transported by the frame. Chang et al. [3] also had the same conclusion: the best strategy to decide which frame to block among two frames being tied, is to block the biggest frame.

Finally, Figure 2c compares our best evaluation function results *DepDropBig* with the results using the Distortion Chain model. We observe that *DepDropBig* results are close to the Distortion Chain model results. This result demonstrates the efficiency of our *DepDropBig* evaluation function. Please note that the Distortion Chain results are always under those for *DepDropBig*. This can be explain by two facts: first the Distortion Chain is a model that estimates the distortion but it was only validated for AVC videos, not HEVC; second the Distortion Chain model estimates the distortion using the CMSE not using the MS-SSIM.

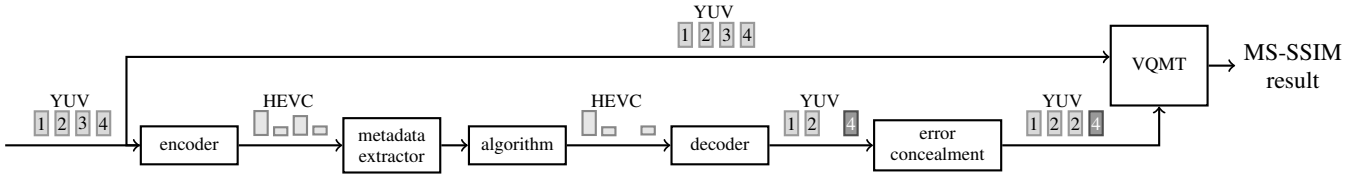
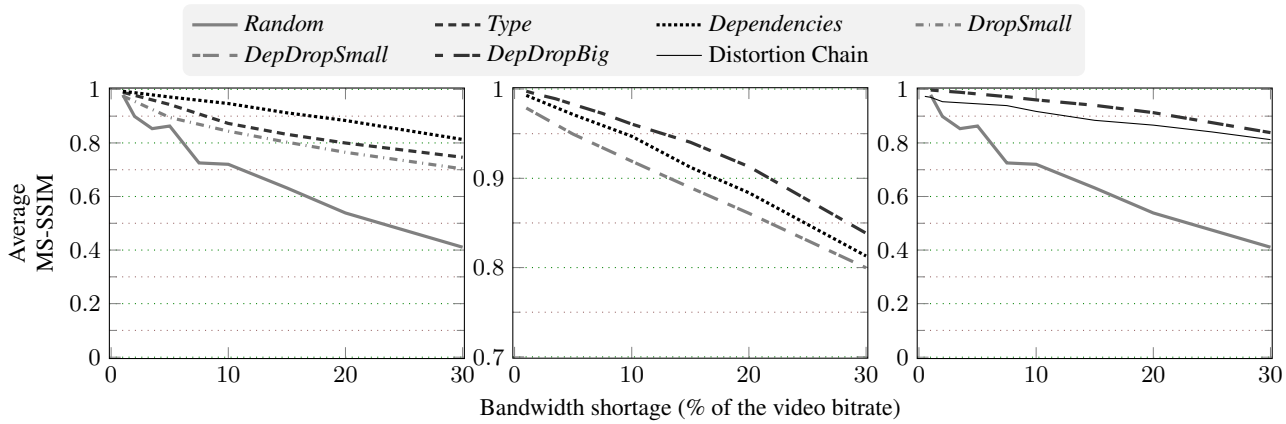


Fig. 1. Testbed: In this example, the algorithm decides to block the packets related to the third frame of the HEVC video. It results in a missing third frame and a decoding problem on the fourth frame in the YUV video. The error concealment adds another second frame to replace the third frame, but it cannot fix the decoding problem on the fourth frame.



(a) One active indicator (b) Two active indicators (c) Comparaison with the Distortion Chain model

Fig. 2. Influence of the different indicators (the legend applies for all three sub-figures)

7.2. Evaluation on New Encoding Structures

We now deal with some recent trends regarding encoding settings. We study the impact of the GOP structure on the blocking strategy as introduced in Section 6.2.

For the dynamic scene cut option (Figure 3a) and the 60 fps (Figure 3b), we observe similar results as in Section 7.1. *DepDropBig* and the Distortion Chain results are close to each other and the gap between *DepDropBig* and random packet losses is significant (0.9 instead of 0.5 for 10 % bandwidth shortage).

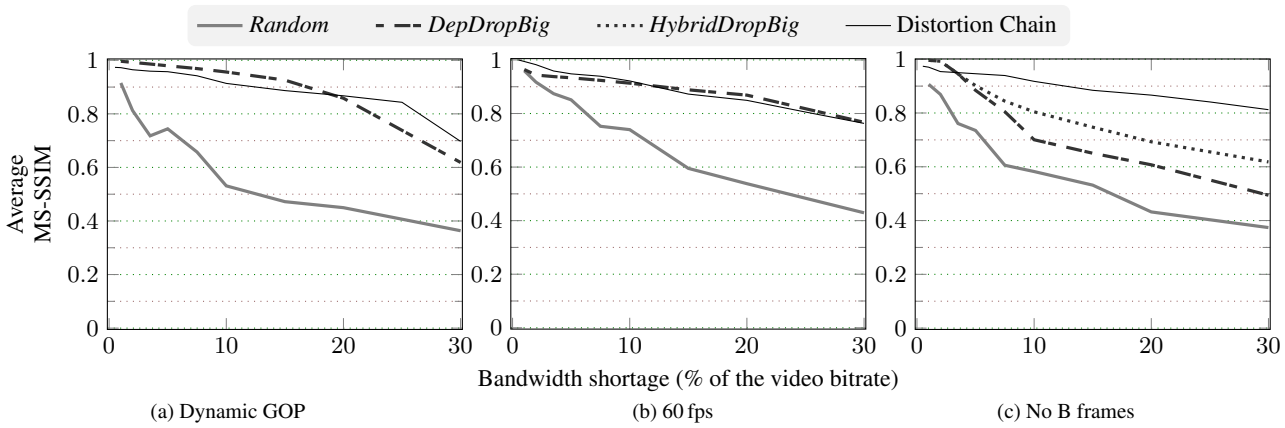
The results for the videos with the no-delay option (Figure 3c) frames are different. Indeed, in this scenario, the videos have no B frames and the frame dependency structure is simple. Each frame is only needed by at most one other frame. For the first 2 % of loss, the frame with zero dependencies are blocked (the last P in each block). But for a loss ratio greater than 2 %, all remaining frames in the video have the same dependencies indicator and so *DepDropBig* selects the biggest frame to be blocked without any distinction between I frames and P frames. By introducing a distinction between I frames and P frames in *HybridDropBig*, we preferentially block P frames, which improves the results because no I frames are blocked. Finally we observe that the gap between the Distortion Chain model results and *HybridDropBig* is significant, which means that our blocking strategies are not efficient for videos without B frames.

8. CONCLUSION

We studied lightweight packet filtering strategies to decide which frames to block in the case of network bandwidth shortage. The goal is to minimize the QoE degradation for the end-user while using only video metadata (frame type, dependencies, and size). We show that the blocking strategies that use a combination of these video metadata are as efficient as more complex distortion estimation strategies. In particular, the best strategy is to block the frames with the smallest number of direct dependencies and, among them, the biggest frames. However, the efficiency of our lighweight strategy drops when the video is encoded with the *no delay* option (without B frames and more linear GOP).

Our paper is the first step of a richer research line. The efficiency of lightweight blocking strategy opens perspectives for their integration within large-scale content delivery technologies. One of our future studies will be to add a mechanism that blocks some frames in an adaptive bit-rate streaming during transient bandwidth drops. Indeed, the adaptive algorithms sometimes over-reacts to a few seconds of bandwidth drop, for example abrupt switch to a much lower-quality video representation or re-buffering with another representation. Instead, blocking frames can be an option to keep on streaming at the same apparent quality and to save some time before a reaction. Such frame blocking option requires predicting throughput for the next seconds and extracting the video metadata of the segment live.

Another future study is to design blocking strategies for video encoded with *no-delay* option. We typically aim at providing the right algorithm for the streaming server of a cloud gaming service. Today, the streaming server can use rate control mechanisms to find a match between the video bit-rate and the actual bandwidth, but switching from one bit-rate to another requires some time at the video encoder. A blocking strategy can enable a smooth transition from one bit-rate to another.



(a) Dynamic GOP (b) 60 fps (c) No B frames

Fig. 3. Comparison of different GOP structures (the legend applies for all three sub-figures)

References

[1] J. Chakareski and P. Frossard. Rate-distortion optimized distributed packet scheduling of multiple video streams over shared communication resources. *IEEE Trans. Multimed.*, 8 (2):207–218, 2006.

[2] J. Chakareski, J. Apostolopoulos, W.-t. Tan, S. Wee, and B. Girod. Distortion chains for predicting the video distortion for general packet loss patterns. In *Proc. of IEEE ICASSP*, 2004.

[3] Y.-L. Chang, T.-L. Lin, and P. C. Cosman. Network-based H. 264/AVC whole-frame loss visibility model and frame dropping methods. *IEEE Trans. Image Process.*, 21(8):3353–3363, 2012.

[4] EPFL's VQMT software. http://mmspg.epfl.ch/vqmt. Accessed on: Nov. 2015.

[5] L. Hu and H. Wildfeuer. Use of content complexity factors in video over ip quality monitoring. In *Proc. of IEEE QoMEX*, 2009.

[6] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In J. W. Byers, J. Kurose, R. Mahajan, and A. C. Snoeren, editors, *Proc. of ACM SIGCOMM IMC*, 2012.

[7] H. Kellerer, U. Pferschy, and D. Pisinger. Introduction to NP-Completeness of knapsack problems. In *Knapsack Problems*, pages 483–493. Springer, 2004. ISBN 978-3-642-07311-3.

[8] Y. J. Liang, J. G. Apostolopoulos, and B. Girod. Analysis of packet loss for compressed video: does burst-length matter? In *Proc. of IEEE ICASSP*, 2003.

[9] T.-L. Lin, S. Kanumuri, Y. Zhi, D. Poole, P. C. Cosman, and A. R. Reibman. A versatile model for packet loss visibility and its application to packet prioritization. *IEEE Trans Image Process.*, 19(3):722–735, 2010.

[10] A. Liotta, D. C. Mocanu, V. Menkovski, L. Cagnetta, and G. Exarchakos. Instantaneous Video Quality Assessment for Lightweight Devices. In *Proc. of ACM MoMM*, 2013.

[11] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis. CQIC: Revisiting Cross-Layer Congestion Control for Cellular Networks. In *Proc. of ACM HotMobile*, 2015.

[12] S. Mehdian and B. Liang. Jointly optimal selection and scheduling for lossy transmission of dependent frames with delay constraint. In *Proc. of IEEE IWQoS*, 2014.

[13] F. Michelinakis, N. Bui, G. Fioravantti, J. Widmer, F. Kaup, and D. Hausheer. Lightweight mobile bandwidth availability measurement. In *Proc. of IFIP Networking*, 2015.

[14] N. Montard and P. Brétillon. Objective quality monitoring issues in digital broadcasting networks. *IEEE Trans Broadcast.*, 51(3):269–275, 2005.

[15] S. Paluri, K. K. R. Kambhatla, B. A. Bailey, P. C. Cosman, J. D. Matyjas, and S. Kumar. A low complexity model for predicting slice loss distortion for prioritizing H.264/AVC video. *Multimed. Tools Appl*, 75(2):961–985, 2016.

[16] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Commun. Surv. Tutor.*, 17(1):469–492, 2015.

[17] N. Staelens, D. Deschrijver, E. Vladislavleva, B. Vermeulen, T. Dhaene, and P. Demeester. Constructing a No-Reference H.264/AVC Bitstream-Based Video Quality Metric Using Genetic Programming-Based Symbolic Regression. *IEEE Trans Circuits Syst Video Techn*, 23(8):1322–1333, 2013.

[18] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *Proc. of IEEE Asilomar Conf. Signals, Systems and Computers*, 2003.

[19] K. Yamagishi and T. Hayashi. Parametric packet-layer model for monitoring video quality of IPTV services. In *Proc. of IEEE ICC*, 2008.