When HTTP/2 Rescues DASH: Video Frame Multiplexing

Mariem Ben Yahia Yannick Le Louedec
Orange Labs, France
firstname.lastname@orange.com

Loutfi Nuaymi Gwendal Simon
IMT-Atlantique, France
firstname.lastname@imt-atlantique.fr

***Abstract*—HTTP Adaptive Streaming is a successful and largely adopted content delivery technology. Yet poor bandwidth prediction, notably in mobile networks, may cause bit-rate oscillations, increased segment delivery delays, video freezes, and may thus negatively impact the end user quality of experience. To address this issue, we propose to exploit the stream prioritization and termination features of the HTTP/2 protocol to achieve video frame filtering and scheduling, so as to maximize the amount of video data received on time by the client. We evaluate with optimal scheduling and filtering algorithms the maximum gain we may expect from such delivery schemes where video frames are carried in dedicated HTTP/2 streams. Evaluation shows that our HTTP/2-based video frame scheduling scheme brings benefits for video quality.**

I. Introduction

The new version of the Hypertext Transfer Protocol (HTTP) standard, namely HTTP/2 [1], is now supported by most of the leading browsers [2]. Among the novelties released in HTTP/2, the feature that enables the server to push content in advance has received the attention of researchers in the multimedia community [10]. However, HTTP/2 also features a novel mechanism to multiplex the delivery of structured data. In this paper, we focus on the benefits one can obtain from using this multiplexing feature for the delivery of video streams over HTTP.

The most popular content providers have recently adopted *adaptive streaming* over HTTP technologies for the delivery of video, whether it is live or on-demand [16]. The adaptive delivery technologies include HTTP Live Streaming (HLS) and the MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard [11]. The video is encoded into multiple video *representations*, which are cut into *segments* of k seconds (k ranges from 2 to 10 seconds). At the end of every segment, the client predicts the available bandwidth for the next k seconds and requests the video representation such that the segment bit-rate matches the throughput. Despite multiple advantages, adaptive streaming technologies present some weaknesses [10]. In particular, these technologies perform well only when the algorithm that predicts the available bandwidth is accurate. Yet, some network environments (*e.g.*, cellular networks) exhibit fast throughput variations, which degrades the prediction accuracy [14]. The main problem occurs when the client over-estimates the available bandwidth for the next k seconds.

No current solution satisfactorily addresses the problem of over-estimated bandwidth predictions in adaptive streaming. When facing the deficit of incoming data, most video players decide to freeze the video play in order to fill the client video buffer. However, the studies from Dobrian et al. [8] as well as the intelligence report from companies [5, 12] identify buffering as the most impacting factor of Quality of Experience (QoE) degradation. Another solution, which is known as *packet filtering* [7], consists in dropping the frames that are the least significant for the video quality. Yet, the implementation of such a solution in HTTP-based multimedia delivery architecture is hard since no control loop exists between the client and the server.

Our proposal is to implement packet filtering techniques by leveraging the concept of HTTP/2 *stream*, and the associated features of "stream reset", "dependencies", and "priorities". HTTP/2 streams enable the client to act on the delivery of structured data. We propose to *send each encoded video frame in a different HTTP/2 stream*. By using the HTTP/2 commands, the client can thus drop video frames by ending an HTTP/2 stream, and give priorities to video frames by setting the priorities of HTTP/2 streams. An advantage of our proposal is the legacy to current DASH platforms: The implementation effort is restricted to the HTTP/2 stack at the client side.

In this paper, we introduce our HTTP/2-based packet filtering delivery scheme. We focus on the theoretical performance gain one can expect when implementing this scheme. We design an Integer Linear Program (ILP), which computes the solution that is obtained by a *clairvoyant* offline algorithm being fully aware of both the future networks variations and the video frames to be delivered. The ILP provides the upper-bound of the achievable gains.

We start the paper by introducing the background on HTTP/2, DASH, and video encoding techniques. We describe our proposal to map HTTP/2 streams with video frames in Section III, and then our optimization models in Section IV. The paper ends with performance evaluation in Section V and conclusion in Section VI.

II. Background

Our proposal exploits specific features from modern video coding techniques, DASH, and HTTP/2. We present them in the following subsections. We then highlight related studies that aim to improve DASH content delivery by leveraging

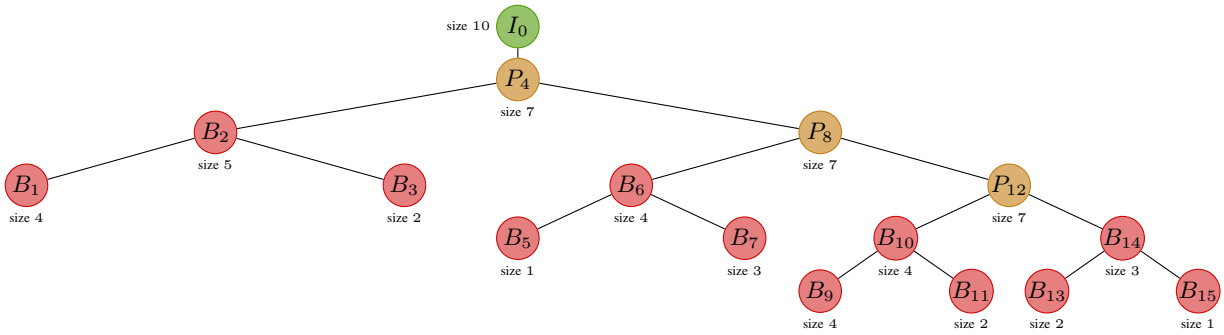

Figure 1: Structure representation of video frames inside GoP $I_0B_1B_2B_3P_4B_5B_6B_7B_8B_9B_{10}B_{11}P_{12}B_{13}B_{14}B_{15}$ in the decoding order with a binary tree. B_2 is a children of P_4 means that B_2 cannot be decoded before the decoding of P_4

these features. Yet, to the best of our knowledge, no previous work has explored video frame filtering by mapping HTTP/2 streams with video frame structure.

A. Video Structure and Video Frame Importance

Video compression uses modern coding techniques, such as H264 and High Efficiency Video Codings (HEVCs), to reduce redundancy in video data. An encoded video is composed of Groups of Pictures (GoPs), each containing intra-coded (I) frames, predicted (P) frames, and bidirectional predicted (B) frames. When a video is delivered over a network to a given client, each video frame has a display time, as well as a decoding deadline, which is the minimum display time of the video frames depending on it. A GoP can be represented in its decoding order, as in Figure 1.

Missing video frames impact the quality of the displayed video. The distortion depends not only on the missing frame (especially its type) but also on its position in the flow of frames. The distortion caused by a set of missing frames close to each other is greater than the sum of the distortion induced by the isolated loss of the same frames [3]. Modeling the distortion function is complex as it is not an additive function of the frames. Yet previous studies investigated simple ways to assign to each video frame several indicators of their relative importance on the display quality. Corbillon et al. [7] proposed to model the importance of video frames as a multi-criteria linear function taking account three indicators: frame type, dependent frames, and frame size. We will consider such a function in the following to decide which frames have to be filtered upon adverse network conditions.

B. Adaptive Streaming

Adaptive streaming technologies, and DASH in particular, enable dynamical adaptation of the video quality to the network conditions. Despite the popularity of DASH [2], some inefficiencies still have to be addressed. Poor bandwidth prediction, notably in mobile networks, may cause bitrate oscillations, increased segment delivery delays, video freezes, and may thus negatively impact the QoE [4, 10, 17].

Since DASH standard enables downloading structured video segments into smaller subsegments [11], some studies manage DASH content delivery at the subsegment level to exploit lower granularity and to provide more flexible content selection and delivery strategies [9, 15]. However, such approaches implemented in HTTP/1 increase the number of HTTP requests made by the client. Our new approach aims to tackle these issues by exploiting native HTTP/2 features.

C. HTTP/2 Structure

The specifications of HTTP/2 was published in May 2015 [1]. As shown in Figure 2, HTTP/2 breaks down the HTTP protocol communication into an exchange of binary-encoded frames. These HTTP/2 frames convey messages that belong to a particular stream and all streams are multiplexed within a single TCP connection.

The order in which the HTTP/2 frames are interleaved becomes a critical performance consideration. Priority among streams can be controlled by setting explicit dependency and weight to streams. The weights set the relative proportion of resources to allocate to sibling streams sharing the same parent stream. In Figure 2, stream S_3 depends on stream S_2 with weight 200 and stream S_4 depends on stream S_2 with weight 100. On the completion of stream S_2, stream S_3 shall get two-third of the resources previously allocated to stream S_2. Dependencies and weights allows the client to provide the server with a "prioritization tree", ensuring optimal delivery of high-priority responses, as shown in Figure 2. Moreover the client can update dependencies and weights at any time, which enables further optimization by reallocating resources in response to user interaction or other signals. Besides, the client can terminate an HTTP/2 stream without closing the underlying TCP connection session.

Huysegems et al. [10] present ten methods to improve the QoE of adaptive streaming based on HTTP/2 *push*, stream multiplexing, stream prioritization, and stream termination features. Yet only the "full-push" approach in which super-short segments are pushed from server to client as soon as they are available is designed and evaluated. The presented techniques involving stream multiplexing, prioritization and termination are applied at the DASH segment level, while our proposal maps HTTP/2 streams with video frames. Most of the other related studies focus on exploiting the HTTP/2 *push*

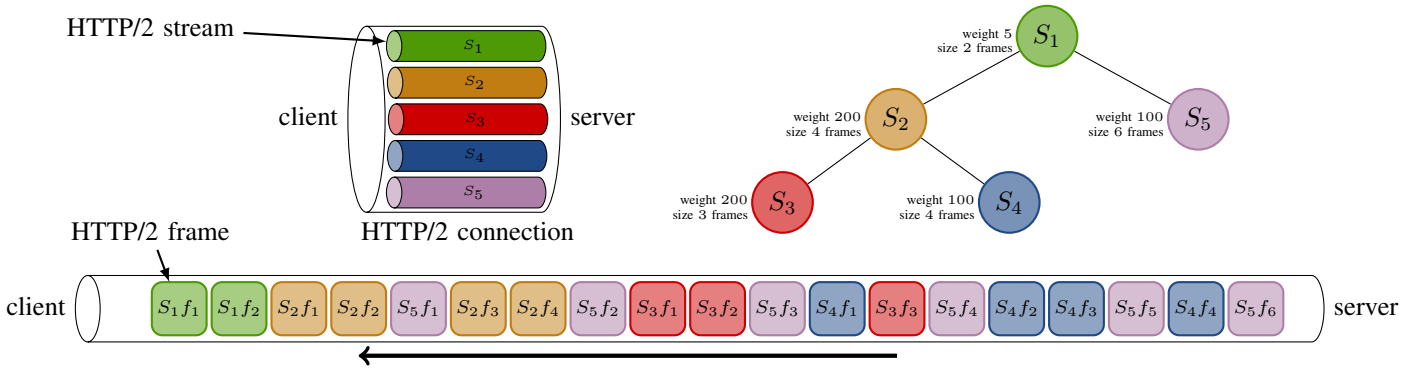

Figure 2: Structure and dependence of the different HTTP/2 streams S_i. Each stream has a weight; the higher is the weight, the higher is the delivery priority. S_2 is son of S_1 means that S_2 cannot be transmitted before the complete transmission of S_1

feature to improve DASH performance [4, 17]. By contrast, our approach exploits the stream prioritization and termination features of HTTP/2.

III. Our Proposal

The intuition behind our proposal comes from observing the similarities between the HTTP/2 stream in Figure 2 and a video frame in Figure 1. Both share some key characteristics: size, dependencies, and priorities. Our proposal is thus to leverage HTTP/2 features related to streams in order to improve the performance of adaptive streaming delivery in the case of over-estimated bandwidth.

A. Handling Video Frame Delivery

We propose to send each video frame in a different HTTP/2 stream so that it is possible to handle the video frames by invoking the commands that are defined in the HTTP/2 standard, including:

Cancel HTTP/2 stream The *rst_stream* command enables the client to cancel the delivery of an HTTP/2 stream, which in our case means to cancel the delivery of a video frame. By using this command, a client decides to not waste throughput for the delivery of a frame that is of low importance or has no chance to arrive on time for the decoder to display it. Note that the *rst_stream* command can be invoked on an HTTP/2 stream while some frames of this stream have already been received.

Set HTTP/2 stream dependencies When the client requests the GoPs for the segment, it associates each video frame to an HTTP/2 stream and it can also set the dependencies between the HTTP/2 streams. The main idea here is to map the HTTP/2 dependencies to the video frame dependencies since it is useless to send a sibling video frame if one of its parent frames is canceled. By respecting the structure of video frame dependencies while attributing each one to the adequate HTTP/2 streams, we do not need to notify the server to cancel the HTTP/2 streams that depend on a canceled HTTP/2 stream. Moreover, the order in which the server delivers the HTTP/2 respects the order of the video frames in the GoP.

Set HTTP/2 streams priorities The *priority* command enables the client to set the priority of a given HTTP/2 stream at any time. In a case of bandwidth shortage, the client can decide to increase the priority of the most important video frames so that the HTTP/2 multiplexing algorithm increases the fraction of the traffic carrying data of these video frames, and consequently increases the probability that these important video frames arrive at the decoder on time. The client can also invoke this command when a video frame is nearly to be decoded but the decoder still misses some HTTP/2 frames of the associated HTTP/2 stream.

B. Main Principles

Our proposal is entirely *client-based*, which is conform to the general architecture of DASH. When the client requests a video segment, it analyzes the structure of the different GoPs and it extracts from the video metadata the *byte ranges* of the frames within each GoP (typically with the technique introduced by Houze et al. [9]). Then the client associates each frame to an HTTP/2 stream with the same dependencies as in the GoP. Finally, the client computes the importance of each frame (for example based on the algorithm presented by Corbillon et al. [7]) and sets the weight of each HTTP/2 stream with the computed video frame importance.

When the DASH selection algorithm inaccurately predicts the bandwidth, and especially when it over-estimates the available bandwidth, several triggers notify the client of troubles in the video delivery. Notable triggers include an abrupt reduction of the amount of data in the buffer and the reduction of the delay between the first reception of the data of a given frame and the playout deadline of this frame.

A client being notified of a video delivery trouble can decide to act on the delivery by using the HTTP/2 commands. Many algorithms of packet filtering can be designed so that the least important frames are dropped in order to ensure the delivery of the most important frames. In this paper, our goal is not to discuss the multiple options for these packet filtering

Input	Meaning
H	Set of HTTP/2 frames successfully received by the client
V	Set of video frames
P^v	Set of video frames on which the video frame v depends
C^v	Set of video frames depending on video frame v
s^v	Number of HTTP/2 frames needed to transmit video frame v
$t^v_{display}$	Display time of video frame v
g^v	Relative importance of video frame v on the entire video quality
h_{size}	HTTP/2 frame size
v_{size}	Video frame v size
t^h_{rx}	Arrival time of HTTP/2 frame h at the client side
M	Display time of the latest video frame of the GOP
Decision	**Meaning**
x^v_h	equal to 1 if HTTP/2 frame h contains data relative to video frame v and 0 otherwise
y^v	equal to 1 if video frame v is displayed (*i.e.*, received as well as all its parents before $t^v_{display}$) and 0 otherwise
z^v	equal to 1 if video frame v is received at any time at the client side, and 0 otherwise (only used in the case of hypothesis H2)

Table I: Notations for inputs and decision variables

algorithms. We rather study optimal frame schedulers, which allows us to determine, for a given configuration, the best scheduling solution. These algorithms are offline and cannot be implemented in real systems, but they provide a bound of the maximum possible gains in specific network conditions with HTTP/2 structure.

IV. Theoretical Optimization Problem

Offline optimal schedulers have a complete knowledge of the network behavior and the video content. Mehdian and Liang [13] present an offline algorithm to schedule video frames in a recursive way, without mentioning any concrete implementation of their theoretical model. We present in Section V results based on their model. Here, we model an ILP inspired by Corbillon et al. [6]. We present this model in the following, with a summary of the notations in Tab I.

A. Optimization Problem

Our model is based on the transmission of video data using HTTP/2 protocol. The input contains the records of a full, completed transmission, including the time at which each HTTP/2 frame of the bit-stream is successfully received at the client side. The algorithm decides which video frame should be sent in each HTTP/2 stream in order to maximize the overall importance of decoded frames.

1) Inputs: We denote by H the set of HTTP/2 frames that are successfully received by the client for the whole transmission. Each HTTP/2 frame $h \in H$ is characterized by its reception time at the client side, denoted by t^h_{rx}. We denote by V the set of video frames that have to be streamed from one server to a client. Each video frame $v \in V$ should be played by the application at the client side at a given time, denoted by $t^v_{display}$. The transmission of a video frame v requires the transmission of s^v different HTTP/2 frames. We consider packet padding if the size of the video frame is not a multiple of the maximum HTTP/2 frame size denoted by $h_{size}, h \in H$. We assume that the HTTP/2 frames size is chosen so that the padding is minimum. For real implementation, an HTTP/2 frame size can be variable with regard to the data it contains.

Regarding the different implementations of the decoder, we distinguish two hypothesis.

- **Hypothesis (H1)**: A video frame v can be decoded and displayed only if all of its parents video frames (*i.e.*, frames v depends on) have been received and displayed on time.
- **Hypothesis (H2)** : A video frame v can be decoded and displayed only if all of its parents video frames have been received before $t^v_{display}$ but not necessarily displayed before its own deadlines.

We denote by P^v the set of video frames parents of a video frame v. We denote by C^v the set of video frames children of a frame v. ($v \in P^{v_c}$ means that $v_c \in C^v$ and v_c cannot be decoded before v is decoded). With regards to the impact of the video frames on the video distortion, we denote by g^v the relative importance of the video frame v [7].

2) Decision variables: The main decision variable of our model x^v_h indicates whether HTTP/2 frame h transports data for video frame v. This decision variable can be interpreted as follows: Each HTTP/2 frame transports data related to a given video frame. A video frame is received if it is transported by s^v HTTP/2 frames. If we consider hypothesis (H1), a video frame v is successfully received, decoded and displayed if:

- s_v HTTP/2 frames carrying data related to v are sent.
- The latest of these frames arrives before $t^v_{display}$
- All video frames in P^v have been displayed.

The decision variable $y^v, v \in V$ is equal to 1 whether the video frame is displayed in time.

If we consider the hypothesis (H2), a video frame that did not arrive on time to be displayed can still be useful for successive frames that depend on it. Then, with (H2), a video frame v is successfully received, decoded and displayed if:

- s_v HTTP/2 frames carrying data related to v are sent.
- The latest of these frames arrives before $t^v_{display}$.
- All video frames in P^v are received before $t^v_{display}$.

For hypothesis (H2), we thus add another decision variable z^v, which indicates whether the video frame v is sent by the server and received by the client at any time.

3) Optimization objectives: We aim to maximize the importance of the successfully received and displayed video frames in order to improve the video quality, Formally:

$$\max \sum_v g^v y^v$$

4) Integer problem modeling: The problem can be modeled differently according to the decoders implementation as described in the hypothesis H1 and H2 bellow. The formulation of ILPs for both hypothesis is as follows.

a) With hypothesis H1:

$$\max \quad \sum_{v} g^v y^v$$

$$\text{s.t} \quad \sum_{h \in H} x_h^v = s^v y^v \tag{1}$$

$$\sum_{v \in V} x_h^v \leq 1 \tag{2}$$

$$y^{v_c} \leq y^v \tag{3}$$

$$\text{if } t_{display}^v \leq t_{display}^{v_p}$$

$$\text{then } t_{rx}^h x_h^{v_p} \leq t_{display}^{v_p}(1 - y^v) + y^v t_{display}^v \tag{4}$$

$$\text{if } t_{rx}^h \geq t_{display}^v$$

$$\text{then } x_h^v = 0 \tag{5}$$

$$\text{With} \quad y^v \in \{0,1\}, x_h^v \in \{0,1\},$$

$$v \in V, h \in H, v_c \in C^v, v_p \in P^v$$

Constraint (1) means that every successfully sent and displayed video frame v must use exactly s^v HTTP/2 frames. Constraint (2) means that every HTTP/2 frame h delivers data related to a unique video frame v. Constraint (3) means that all the frames on which v depends must be also successfully sent and displayed. Constraint (4) means that all the parent video frames $v_p \in P^v$ relative to a video frame $v \in V$ have to been received before the display time of v if v is successful. Constraint (5) means that data of v is sent in a HTTP/2 frame h only if h is received before the video frame display time.

b) With hypothesis H2:

$$\max \quad \sum_{v} g^v y^v$$

$$\text{s.t} \quad \sum_{h \in H} x_h^v = s^v z^v \tag{6}$$

$$\sum_{v \in V} x_h^v \leq 1 \tag{7}$$

$$y^v \leq z^{v_p}, v_p \in P^v \cup \{v\} \tag{8}$$

$$x_h^v t_{rx}^h \leq t_{display}^{v_c} + M(1 - y_{v_c}), v_c \in C^v \cup \{v\}) \tag{9}$$

$$\text{With} \quad z^v \in \{0,1\}, y^v \in \{0,1\}, x_h^v \in \{0,1\},$$

$$v \in V, h \in H, v_c \in C^v, v_p \in P^v$$

Constraint (6) means that every sent video frame v must use exactly s^v HTTP/2 frames. Constraint (7) means that every HTTP/2 frame h delivers data related to a unique video frame v. Constraint (8) means that all the frames on which v depends must be successfully sent. Constraint (9) means that all the parent video frames $v \in P_c^v$ relative to a video frame $v_c \in V$ must be received before the display time of v_c if v_c is successfully displayed. It also means that one necessary condition for frame v to be displayed is to be entirely received before its display time.

For both hypothesis (H1) and (H2), similar problems were studied in [14]. The author demonstrated their NP-hardness, proposed several heuristics and evaluated their performances.

V. Tests and evaluation

Our goal is to evaluate the best achievable performance of our proposal. We use the model presented in Section IV and we show the results of the optimal solution against a neutral scheduler for various configurations (video and networks). We thus show the benefits one can expect from implementing our HTTP/2-based frame filtering solution.

A. Dataset

We use logs of real TCP connections, which include, for each TCP segment, the transmission and reception timestamps, as well as its playload size. We capture traces from a mobile device having a network connection limited to 4 Mbps. We then extract the number of TCP segment packets and bytes exchanged. The median duration, number of TCP segments and volume of downloaded data on the TCP transmission captures are about 20 seconds, 15,000 segments, and 20 MB respectively. By using this TCP dataset, we set an average HTTP/2 frame size to model the HTTP/2 traffic containing the transmission and reception timestamps of each HTTP/2 frame. We include in our model an initial buffering delay ranging from 0.1 to 0.5 seconds to emulate different initial video playing conditions. Finally, we transmit an HEVC video with 24 frames per second and average bit-rate of 5 Mbps. That is, the client has a deficit of 1 Mbps between the available bandwidth and the video.

B. Results

We show in Figure 3 the ratio of received frames (I, P, and B frames) to the total number of frames. We also present the ratio of the importance of the received frames to the total importance of all frames. The results confirm our intuition that our proposal can significantly improve the QoE of the users. In the worst conditions (initial buffer only 0.1 s), the ratio of received frames is very low when no scheduler is implemented, although the optimal scheduler manages to transmit more than four fifth of the most important frames, which results in an overall ratio of frame importance greater than 0.8. When the initial buffer is larger, the optimal scheduler achieves an importance ratio of 0.94, which, according to Corbillon et al. [7], is a video quality without noticeable distortion.

We also compute the same ratios for the two hypothesis (H1) and (H2) in shorter sequences due to the complexity of the ILP for hypothesis (H2). The results are in Table II. The results confirm that a decoder that accepts frames that arrive too early for the decoding of successive frames perform much better than decoder that ignore the frames that arrive late.

initial buffer	**0.1 s**		**0.2 s**		**0.5 s**	
	H1	**H2**	**H1**	**H2**	**H1**	**H2**
I-frame ratio	0.50	0.50	0.50	0.50	1.00	1.00
P-frame ratio	0.25	0.75	0.25	1.00	1.00	1.00
B-frame ratio	0.38	0.59	0.38	0.65	0.80	0.94
importance ratio	0.35	0.76	0.35	0.81	0.89	0.98

Table II: Performance comparison of optimal schedulers

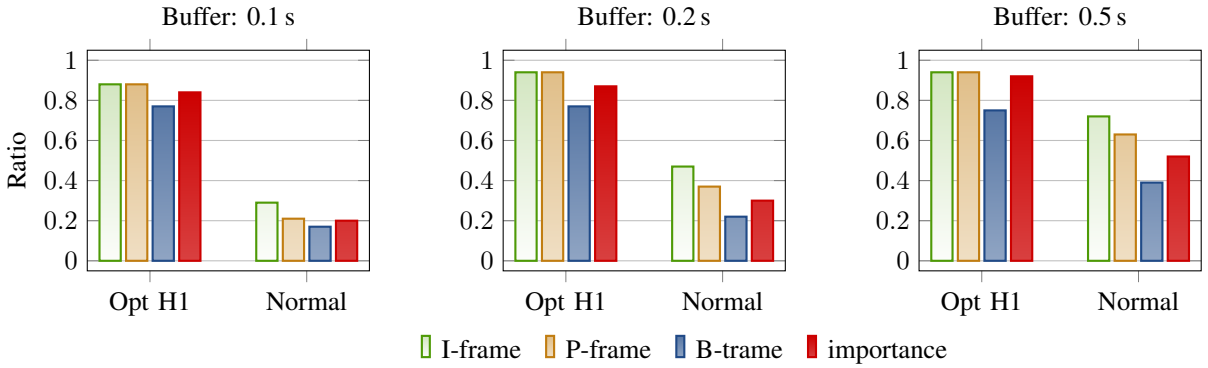

Figure 3: Ratio of received frames to the total frames in the video and ratio of the overall importance of the received frames to the total importance of all frames. The optimal scheduler with hypothesis H1 and the neutral scheduler without any priority

VI. Conclusion

This paper introduces an HTTP/2 based video delivery scheme. We focus on the problem of over-estimated bandwidth in DASH. We propose to map HTTP/2 stream to video frame to enable a better video data scheduling. We show that the implementation of this mapping is possible with reasonable development. To study the outcome of this idea, we design theoretical models that compute the optimal scheduling solution for two decoder implementation. The results demonstrate two main statements: (i) the gap between the current scheduler performances and the optimal solution over HTTP/2 is significant and (ii) a decoder that accepts frames arriving after their display time but early enough for decoding successive frames improve delivery. Many perspectives are open for future works, including a performance evaluation of various practical schedulers using HTTP/2 commands, and best combination of frame filtering solutions and video representation switching.

References

[1] M. Belshe, R. Peon, and M. Thomson. Hypertext transfer protocol version 2 (http/2) standard. Technical Report RFC-7540, IETF, May 2015.

[2] P. Campbell. Why every brand needs to migrate to http/2 and how to do it. Technical report, State of Digital, May 2016. http://tinyurl.com/zefhtkm.

[3] 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.

[4] W. Cherif, Y. Fablet, E. Nassor, J. Taquet, and Y. Fujimori. Dash fast start using http/2. In *ACM NOSSDAV*, 2015.

[5] Conv. Viewer experience report 2015. Convivia Annual Report, December 2015. http://tinyurl.com/hc5nwdy.

[6] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon. Cross-layer scheduler for video streaming over MPTCP. In *ACM MMSys*, 2016.

[7] X. Corbillon, F. Boyrivent, G. A. D. Williencourt, G. Simon, G. Texier, and J. Chakareski. Efficient lightweight video packet filtering for large-scale video data delivery. In *IEEE ICME Packet Video*, 2016.

[8] F. Dobrian, A. Awan, D. A. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang. Understanding the impact of video quality on user engagement. *Commun. ACM*, 56(3):91–99, 2013.

[9] P. Houze, E. Mory, G. Texier, and G. Simon. Applicative-layer multipath for low-latency adaptive live streaming. In *IEEE ICC*, 2016.

[10] R. Huysegems, T. Bostoen, P. Rondao-Alface, J. van der Hooft, S. Petrangeli, T. Wauters, and F. D. Turck. Http/2-based methods to improve the live experience of adaptive streaming. In *ACM Multimedia Conf. MM*, 2015.

[11] ISO. Dynamic adaptive streaming over HTTP (DASH). Technical Report 23009, ISO/IEC, 2012.

[12] Limelight. The state of online video. Limelight Annual Report, December 2016. http://tinyurl.com/hw4x4b3.

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

[14] A. Samba, Y. Busnel, A. Blanc, P. Dooze, and G. Simon. Instantaneous throughput prediction in cellular networks:which information is needed? In *IFIP/IEEE IM Conf.*, 2017.

[15] V. Swaminathan and S. Wei. Low latency live video streaming using http chunked encoding. In *ACM MMSP*, 2011.

[16] N. Weil. The State of MPEG-DASH 2016. Technical report, Streaming Media, March 2016. http://tinyurl.com/hs2zbuh.

[17] M. Xiao, V. Swaminathan, S. Wei, and S. Chen. Evaluating and improving push based video streaming with http/2. In *ACM NOSSDAV*, 2016.