SCHOOL OF ENGINEERING - STI
SIGNAL PROCESSING INSTITUTE
Jean-Paul Wagner and Pascal Frossard

CH-1015 LAUSANNE

Telephone: +41 21 693 47 09
Telefax: +41 21 693 7600
e-mail: {jean-paul.wagner,pascal.frossard}@epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

JOINT PLAYBACK DELAY AND BUFFER OPTIMIZATION IN SCALABLE VIDEO STREAMING.

Jean-Paul Wagner and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Signal Processing Institute Technical Report

TR-ITS-2007.13

December 20th, 2007

Part of this work has been submitted to IEEE Transactions on Circuits and Systems for Video Technology.
This work has been partly supported by the Swiss National Science Foundation.

Joint Playback Delay and Buffer Optimization in Scalable Video Streaming

Jean-Paul Wagner and Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute - LTS4
CH-1015 Lausanne
{*jean-paul.wagner, pascal.frossard*}*@epfl.ch*

***Abstract*—This paper addresses the problem of the transmission of scalable video streams to a set of heterogeneous clients through a common bottleneck channel. The packet scheduling policy is typically crucial in such systems that target smooth media playback at all the receivers. In particular, the playback delays and the transmission strategy for the packets of the different layers have to be chosen carefully. When the same video is sent simultaneously to multiple clients that subscribe to different parts of the stream, the playback delay cannot be jointly minimized for all the clients. We therefore propose delay optimization strategies along with low complexity solutions for a fair distribution of the delay penalty among the different receivers. Once the delays are selected, we show that there exists a unique scheduling solution that minimizes the buffer occupancy at all the receivers. We derive an algorithm for computing the optimal sending trace, and we show that optimal scheduling has to respect the order of the packets in each media layer. Interestingly enough, solving both delay and buffer optimization problems sequentially leads to a jointly optimal solution when the channel is known. We finally propose a simple rate adaptation mechanism that copes with unexpected channel bandwidth variations by controlling the sending rate and dropping layers when the bandwidth becomes insufficient. Experimental results shows that it permits to reach close to optimal performances even if the channel knowledge is reduced. Rate adaptation provides an interesting alternative to conservative scheduling strategies, providing minor and controllable quality variations, but with a higher resulting average quality.**

I. Introduction

Due to the rapid evolutions in consumer electronics, the possibility to adapt to client preferences or to customize services becomes predominant in multimedia applications. We consider in this paper the problem of the simultaneous delivery of a scalable media stream to heterogeneous clients that present different computing capabilities, different access bandwidths, or different user requirements. Each one of these clients selects to receive an appropriate subset of scalability layers, such that the received stream is decodable by the client and satisfies its scalability needs. A typical example of such a system is given in Figure 1, where a streaming server connects to heterogeneous clients directly or through a streaming proxy and broadcasts a stored scalable media stream. Scalable video streaming systems have generally to respect a bottleneck channel bandwidth, which prevents the immediate delivery of the media data to all the clients. This limitation may be imposed by channel or disk bandwidth constraints, admission control or pre-determined traffic specifications (e.g., TSPEC in the 802.1e wireless standard). Client buffering capabilities may help to smooth the discrepancies between the video source rate and the available bandwidth with a sustained quality, at the price however of an increased playback delay at the client. It becomes therefore important to derive efficient packet scheduling strategies such that smooth playback can be ensured at each client and that the overall quality of service is maximized.

This work has been partly supported by the Swiss National Science Foundation.

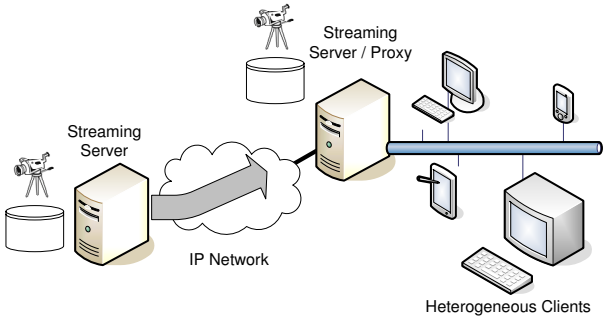

Fig. 1. Example of a scalable video streaming system.

Several works have studied the problem of efficient packet scheduling in different streaming scenarios. The problem of minimizing the playback delay and buffering needs for a single receiver and non-scalable streams under guaranteed rate constraints has been previously addressed in [1]–[3]. The problem has been nicely formalized in more general terms in [4] and [5]. In [6] the authors discuss optimal streaming of layered video under random bandwidth models, when the buffer is not constrained at the decoder. The error concealment at the decoder is further considered in [7] where the scheduling decisions are based on a Markov Decision Process to cope with unpredictable bandwidth variations. The authors of [8] address a similar scenario, where the number of layers that are transmitted are computed from local decisions based on expected run-time estimation. None of the above work however considers the problem of multiple clients that participate together to the streaming session.

A scheduling algorithm that minimizes the buffer occupancy of a single client that receives a single stream has been proposed in [9]. Our work extends this algorithm to provide jointly minimal buffer occupancy at heterogeneous clients that decode different subsets of the same layered stream. Optimal multiplexing for continuous media streaming is discussed in [10]. However the authors focus on bandwidth efficiency and do not discuss the delay nor the buffer occupancy experienced by a client. Layered video streaming has been studied in relation with multicast delivery schemes in [11], [12], without addressing the specific problem of heterogeneous receivers with delay and buffer constraints. None of the cited papers addresses the problem of multiplexing a layered video stream onto a broadcast channel by targeting on one hand a set of minimal playback delays for heterogeneous clients, and at the other hand the minimum buffer occupancy at each one of these clients.

In this paper, we propose to optimize the selection of the playback delays for the different clients in order to have a fair distribution of the delay penalty induced by the broadcast-like media transmission. We

show that the minimal playback delays cannot be jointly achieved for all the clients, and we derive low cost optimization algorithms for computing playback delay sets under different client prioritization policies. Once the playback delays are given, we prove that minimum buffer occupancy can be simultaneously attained for all the clients. There is moreover a unique sending trace that attains the optimal solution in this case, and we propose an algorithm that implements the optimal transmission of the packets from the different layers. When both optimization problems are solved sequentially, the system can design a mechanism that jointly optimizes delays and buffers. To the best of our knowledge, this work is a first effort to address the playback delay optimization problem, together with the buffer minimization problem for broadcast to heterogeneous clients. Finally, we show how the optimal scheduling solution can be modified with a simple adaptive rate control algorithm when the knowledge about the channel bandwidth is limited. This solution provides an interesting alternative to conservative scheduling schemes in some practical scenarios with unpredictable bandwidth, while it offers an improved average quality but minor and controllable quality variations.

The paper is organized as follows: we provide an overview of the system under consideration and discuss media scheduling properties in Section II. We present the delay optimization solutions in Section III, and we analyze the buffer minimization problem in Section IV. Section V introduces an adaptive rate control algorithm to cope with unexpected bandwidth variations and discusses its performance in practical scenarios.

II. Scalable Video Broadcast

A. System Overview

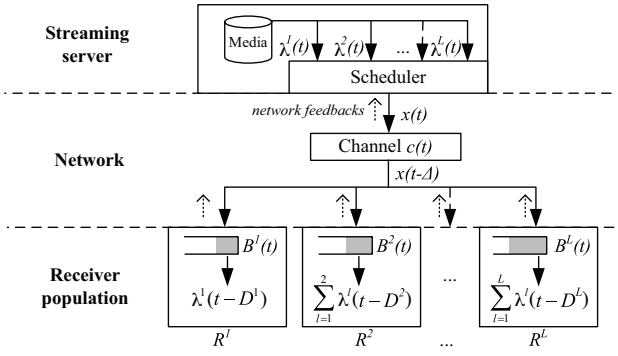

Fig. 2. Formal view of the system.

We present an overview of the system under consideration in this paper. A formal representation of the system is given in Figure 2. A streaming server sends a scalable media stream to a population of receivers through a bottleneck channel. This bottleneck represents for example a shared channel or network segment with limited bandwidth, or the disk bandwidth limitations in a video-on-demand server. The bottleneck channel is given by its bitrate $c(t)$, which indicates how many bits the channel is able to transmit at any time t, and possibly by a maximum network latency Δ.

Generally, the server's knowledge about the channel availability is extracted from client or network feedback. In this paper we will assume perfect channel knowledge at the server, which leads to an upper bound on achievable performance for any predictive scheme, where the server estimates the available channel. In particular, this assumption is verified for constant bit rate channels or when the bandwidth is controlled by deterministic guarantees (e.g., TSPEC in the recent 802.11e wireless protocol). In the rest of this paper, the channel rate $c(t)$ is the rate available for the broadcast application, and we do not limit the study to any particular congestion control or rate allocation strategy.

The scalable video stream is built on several hierarchical layers. Each of the L layers is completely determined by its source or playout trace $\lambda^l(t), 1 \leq l \leq L$, which indicates the size of the layer l at time t. When a hierarchy exists between video layers, the decoding of layer l is made contingent on the correct decoding of all inferior layers, from 1 up to $l-1$. Batches of clients simultaneously access the same video sequence, possibly with different scalability levels. The receivers are grouped together into L sets, based on the number of video layers or the resolution that they have requested. We denote as R^l, $(1 \leq l \leq L)$ the set of clients that receive all layers up to the l^{th} layer.

After the first bit is sent by the server, each receiver in R^l buffers the video data for a *playback delay* D^l. The video bits are stored in the receiving buffer, whose content at any time is further denoted as $B^l(t)$. After the initial playback delay, the receiver decodes and plays continuously a video whose resolution corresponds to the series of additive layers it has requested. The playback delay can be different for each group of receivers R^l. However, the set of playback delays $\mathcal{D} = \{D^l\}_{l=1}^{L}$ should be chosen such that non-disruptive playback of the sequence can be achieved for any set of receivers R^l, i.e., such that no buffer underflow occurs at any receiver [1]. At the same time, the choice of the playback delay impacts the quality-of-service perceived by the end-user. It therefore requires an efficient packet scheduling strategy in order to reach a proper trade-off between user experience and resilience to underflows and bandwidth limitations. Before addressing the problems of playback delay selection, we describe below the problem of packet scheduling in media streaming applications that generally impose strict timing constraints.

B. Media Scheduling Formalism

We now describe in more details the packet scheduling characteristics in media streaming applications. When the channel is constrained, the streaming server has generally to implement effective scheduling algorithms, in order to ensure timely delivery of media packets and to avoid buffer underflow at receivers. The packet transmission strategy is chosen in order to meet criteria such as desired distortion or delay [13], [14], or maximum utilization of the available channel bitrate. The packet scheduler outputs a stream with a *sending rate* $x(t) \leq c(t), \forall t$ that indicates the number of bits fed into the channel at any time instant t.

We denote the cumulative source, sending and channel rate functions with capital letters (S, X, C), where a cumulative rate function is defined as the total number of bits that have been counted since time $t = 0$. For example, $C(t) = \int_0^t c(u)du$ is the number of bits the channel can transmit up to time t. Note that the cumulative rate functions are all wide-sense increasing in t. We further define $s_D(t)$ as the number of bits consumed by the decoder that starts playing video after a playback delay D. It is written as:

$$s_D(t) = \begin{cases} 0 & , 0 \leq t < D \\ s(t-D) & , t \geq D \end{cases}$$

$S_D(t) = S(t-D)$ is the corresponding cumulative function.

The scheduler has to ensure that the client does not experience any buffer starvation, when the video decoding starts after a playback delay that has been selected a priori. Figure 3 illustrates the importance of the playback delay for smooth video decoding in a scenario with

[1] In the remaining of the paper, we use R^l to design a set of receivers that subscribe to the resolution level l or one of the receivers in this set, interchangeably.

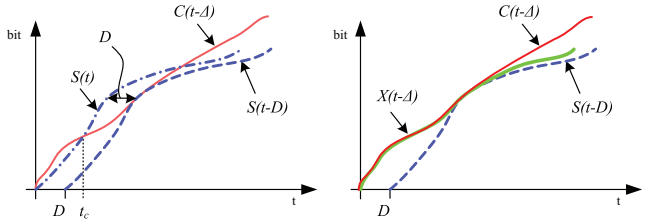

Fig. 3. *Left:* Playback delay and buffer underflow prevention. *Right:* Schedulable play-out trace and a corresponding sending rate trace.

a single client. If the client starts playback at the reception of the first bit, a buffer underflow occurs at time t_c. Starting playback at the client after D time units makes sure that the buffer underflow does not occur. We say that a source trace $s(t)$ is *schedulable* over a channel with available bandwidth $c(t)$, with a playback delay D, if the following *schedulability condition* holds for all t:

$$S_D(t) \quad \leq \quad C(t-\Delta) \tag{1}$$

If the condition (1) is met, this implies that the server can find a scheduling solution or equivalently a sending trace $x(t)$ such that each of the following necessary conditions are satisfied for all t:

$$S_D(t) \quad \leq \quad X(t-\Delta) \tag{2}$$

$$x(t) \quad \leq \quad c(t). \tag{3}$$

The first inequality makes sure that the server transmits a number of bits that is sufficient for decoding the video stream at all time instants t, with a playback delay D and a maximum network latency Δ. As a constant value of Δ implies a simple time shift in each of the above inequalities, we will consider that $\Delta = 0$ for the sake of clarity and without loss of generality in the remainder of this paper. The second condition simply imposes that the number of bits transmitted by the server at any time instant is not larger than the channel rate. Note that the latter condition implies $X(t) \leq C(t), \forall t$, but that the reverse is not true. If the playback delay D is chosen such that the above conditions hold, a scheduling solution can be found. Each valid scheduling strategy generates a sending rate $x(t)$ that satisfies Eqs. (2) and (3) for all t. Finally, the buffer occupancy of a media client that receives a data rate $X(t)$ and plays out a sending trace $S_D(t)$ is given by:

$$B(t) \quad = \quad X(t) - S_D(t), \forall t. \tag{4}$$

The above equation shows that the buffer occupancy is dependent on the choice of the playback delay, so that the joint minimization of both components becomes non-trivial. We first present a method to optimize the choice of the playback delays in scalable streaming systems. Then we show how to define a scheduling strategy that minimizes also the buffer occupancy.

III. Playback Delay Optimization

A. Preliminaries

In this section, we discuss the choice of the playback delays for the different sets of clients R^l that simultaneously receive the scalable stream. Small playback delays usually lead to a better quality of service, and we will present algorithms for optimizing the choice of playback delays, under different metrics. We first give a preliminary analysis of the playback delay, and define the minimal playback delay for a client R^l that decodes l layers of the scalable video stream.

We introduce here some general results inspired from [15]. Suppose that we have two increasing non-zero functions $F(t)$ and $G(t)$ such that $\lim_{t\to\infty} F(t) \geq \lim_{t\to\infty} G(t)$. We define the (maximum) horizontal distance between $F(t)$ and $G(t)$ as follows:

$$h(G,F) \quad = \quad \sup_t \left(F^{-1}\left(G(t)\right) - t\right), \tag{5}$$

where $F^{-1}(t) = \min\{t : F(t) \geq x\}$ is a pseudo-inverse of $F(t)$. The following relations hold:

$$h(G,F) = 0 \quad \Leftrightarrow \quad F(t) \geq G(t), \forall t \text{ and} \tag{6}$$

$$\exists \tau \text{ s.t. } F(\tau) = G(\tau) \tag{7}$$

$$h(G,F) < 0 \quad \Leftrightarrow \quad F(t) > G(t), \forall t \tag{8}$$

$$h(G,F) > 0 \quad \Leftrightarrow \quad \exists \tau \text{ s.t. } F(\tau) < G(\tau). \tag{9}$$

When $F(t)$ and $G(t)$ respectively represent the cumulative channel and source traces, the horizontal distance between F and G represents the minimal playback delay that is necessary for a smooth decoding of the source stream. In other words, it represents the minimal shift that as to be applied on G, such that the schedulability condition is verified. Formally, we have the following property.

Property 1: If $h(G,F) > 0$ and $G'(t) = G(t - h(G,F))$, then $h(G',F) = 0$. In other words, $h(G,F)$ is the minimum shift we need to apply on $G(t)$, so that $F(t) \geq G'(t)$, $\forall t$.

With multiple traces, we have also:

Property 2: Let $F(t)$, $G(t)$ and $G'(t)$ be non-decreasing functions such that $G'(t) > G(t)$, $\forall t$. Then: $h(G',F) > h(G,F)$. Indeed by the definition of $h(\cdot)$ and $F^{-1}(\cdot)$, and because $F(t)$ is non-decreasing, the result follows immediately, as $F^{-1}\left(G'(t)\right) > F^{-1}\left(G(t)\right)$, $\forall t$. Similarly, if $G'(t) < G(t)$, $\forall t$ then $h(G',F) < h(G,F)$.

We can therefore define the minimal playback delay D^l_{min} for smooth playback at the receiver R^l. It is given by

$$D^l_{min} = h\left(S^l(t), C(t)\right), \tag{10}$$

where $S^l(t) = \sum_{k=1}^{l} \Lambda^k(t)$ is the cumulative rate of the stream at resolution l. From Property 2, we know that $D^l_{min} \leq D^{l+1}_{min}$, $\forall 1 \leq l \leq L-1$, since the rate traces are positive valued functions, and $S^{l+1}(t) \geq S^l(t), \forall t$. If the layer l is decoded after a minimal playback delay D^l_{min}, the only valid scheduling solutions are strategies where the playback delay for layers $k < l$ is not any larger than D^l_{min}. It is actually not possible to reduce the minimal playback delay for the client R^l, even by changing the scheduling of the lower layer streams.

Let us define $\vec{\delta} = [\delta_1, .., \delta_l]$, with $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_l \geq 0$. We have the following Lemma, which shows that the minimal playback delay required for a smooth decoding of the resolution level l cannot be smaller than D^l_{min}, even if the lower layers are decoded with a smaller delay.

Lemma 1: Consider a set of L non-decreasing functions $\{H^l(t)\}_{l=1}^{L}$ and a non-decreasing function $F(t)$, defined $\forall t$. We have, $\forall l$, $1 \geq l \geq L$:

$$h\left(G^l, F\right) \leq h\left(G^l_{\vec{\delta}}, F\right),$$

where $G^l(t) = \sum_{k=1}^{l} H^k(t)$ and $G^l_{\vec{\delta}}(t) = \sum_{k=1}^{l} H^k(t+\delta_k)$.

Proof: As the functions $\{G^l(t)\}_{l=1}^{L}$ are non-decreasing, we have, $\forall l$, $1 \geq l \geq L$ and $\forall \delta_l \geq 0$: $H^l(t) \leq H^l\left(t+\delta_l\right)$. Thus, $\forall l$, $1 \geq l \geq L$,

$$G^l(t) \leq G^l_{\vec{\delta}}(t) \ , \forall t.$$

From Property 2, it follows that $h\left(G^l, F\right) \leq h\left(G^l_{\vec{\delta}}, F\right)$. ■

From the above results we conclude that any playback delay smaller than D^l_{min} results in a buffer underflow at the receiver R^l, while any larger playback delay allows for decoding without experiencing a buffer underflow. This permits to derive a simple

bisection search algorithm for computing the minimal delay D^l_{min}, similar to Algorithm 1.

Algorithm 1 $D_{min} = getDmin\,(C(t), S(t))$

$D_{low} \Leftarrow 0$
$D_{high} \Leftarrow$some large value
while $(D_{high} - D_{low}) > 1$ **do**
  $D_{test} \Leftarrow \left\lfloor \frac{D_{low}+D_{high}}{2} \right\rfloor$
  if $S\,(t - D_{test}) \leq C(t)$, $\forall t$ **then**
    $D_{high} \Leftarrow D_{test}$
  else
    $D_{low} \Leftarrow D_{test}$
  end if
end while
$D_{min} = D_{test}$

It is important to note here that achieving the minimum playback delay for a given layer l does not necessarily guarantee that a minimum playback delay is also achieved for any other layer. In general, if the transmission strategy is chosen in order to minimize the delay at layer l, without considering any other layer k, with $k < l$, the clients that only subscribe to the lower layers are penalized by a playback delay that might be larger than necessary. If, in the contrary, the scheduler decides to minimize the playback delay for layer k, it generally increases the playback delay any other layer l, with $l > k$. The choice of playback delay therefore results from a typical trade-off between the delays imposed to the different layers, since the delays cannot be minimized simultaneously for all the layers. When the playback penalty is increased for the lower layers, it typically saves channel bits that can be used for decreasing the playback delay penalty for higher layers. In the next section, we formulate an optimization problem for the choice of the playback delays for different policies.

B. Problem Formulation

Consider a channel given by its cumulative rate trace $C(t)$, and a set of L hierarchically coded layers given by their cumulative source rate traces $\{S^l\}_{l=1}^L$. The channel connects a streaming server to L sets of receivers $\{R^l\}_{l=1}^L$, that simultaneously subscribe to layers up to l. Let $\mathcal{D} = \{D^l\}_{l=1}^L$, with $D^1 \leq D^2 \leq \ldots \leq D^L \leq D_{max}$ denote the set of playback delays imposed to the different sets of clients. The joint minimization of the playback delays for all heterogeneous receivers is generally not achievable in broadcast-like scenarios, as discussed above. We therefore formulate the following optimization problem, which targets a fair selection of the playback delays. Let D^l_{min} represents the minimal playback delay that can be offered to the client R^l, when other clients are not considered. A fair distribution of the playback delay among the different clients can be achieved by controlling the penalties $\Delta = [\Delta^1, ..\Delta^L]$, with $\Delta^l = D^l - D^l_{min}$, $1 \leq l \leq L$, in addition to minimizing the playback delays. The playback delays can therefore by chosen as

$$\mathcal{D}_{opt} = \arg\min_{\mathcal{D}} \varphi\left(\{D^l\}, \{\Delta^l\}\right) \tag{11}$$

under the condition that $S^l_{\mathcal{D}}(t) \leq C(t)\ \forall l, 1 \leq l \leq L$, where $S^l_{\mathcal{D}}(t) = \sum_{i=1}^{l} \Lambda^i_{D^i}(t)$, i.e., all the traces are schedulable from Eq. (1). The function φ is a generic cost function that combines the average playback delay and the delay penalty imposed to each layer due to the broadcast-like distribution. Finding the best set of playback delays $\mathcal{D}_{opt}$ is actually a combinatorial optimization problem, and its solution generally implies a full search algorithm. We show in the next sections how the search space can be reduced for solving the generic optimization problem of Eq. (11). We also present efficient solutions to the problems of fair or weighted distribution of the delay penalty Δ^l between the different layers.

C. Reduced search space

In order to solve the joint delay optimization problem, we propose to limit the search space of possible delay values. We know already from the above discussion that the minimal playback delay for clients that decode the stream up to layer l, is D^l_{min}. It corresponds to the lowest achievable delay, when clients R^k with $k \neq l$ are not considered in the delay computation. Generally, the playback delay for layer l is larger than D^l_{min} when the scheduler also tries to reduce the delay for the lower layers $1 \leq k \leq l$. In order to reduce the search space, we are looking for a reasonable upper-limit on the search interval. From Lemma 1, the worse case policy for clients R^l consists in minimizing iteratively the delays for all the clients R^k with $1 \leq k \leq l$. We describe below the greedy delay allocation policy, and we denote the resulting delay D^l_{greedy}.

The greedy delay allocation first minimizes the playback delay of the first layer, which is thus decoded after $D^1_{greedy} = D^1_{min}$. It then iteratively allocates the smallest possible delay to the different layers, given the greedy delay allocation for the lower layers. Formally, we denote the available channel bandwidth for transmitting the layer l as $C^l(t) = C(t) - \sum_{k=1}^{l-1} \Lambda^k(t - D^k_{greedy})$. Therefore, the minimal playback delay for layer l becomes $h(C^l, \Lambda^l)$ under the greedy allocation policy. This scenario results in an upper bound D^L_{greedy} on the playback delay for the highest layer L, when all playback delays are chosen in a greedy manner. In particular, delays that are larger than D^L_{greedy} also provide valid scheduling solutions. However, increasing the playback delay D^L does not reduce the playback delay of the lower layers, and rather contribute to increasing the standard deviation of the penalties given in Eq. (11). The delays obtained by the greedy allocation can therefore be safely considered as the upper-limits of the search intervals. The greedy layered scheduling strategy is shown in Algorithm 2.

Algorithm 2 $\left(\{D^l_{greedy}\}\right) = GreedyD\left(C(t), \{\Lambda^l(t)\}\right)$

$C^1(t) \Leftarrow C(t)$
for $l = 1$ to L **do**
  $D^l_{greedy} \Leftarrow getDmin\left(C^l(t), \Lambda^l(t)\right)$
  $C^{l+1}(t) \Leftarrow C^l(t) - \Lambda^l(t - D^l_{greedy})$
end for

As the greedy delay allocation provides the worst case solution for minimizing the delay for all the receivers, we can limit the search domain for computing the best set of playback delays to the interval $[D^l_{min}, D^l_{greedy}]$, $\forall l$. In addition, due to the hierarchical nature of the scalable video stream, we know the delay can only take non-decreasing values when the number of layers increases (i.e., $D^k \leq D^l$ when $k \leq l$). We can therefore limit the number of potential solutions that need to be tested for optimality by the search algorithm, by setting the condition that $D^L \in [D^L_{min}, D^L_{greedy}]$. Then, for each possible value of D^L, we constrain the search algorithm to test values of D^{L-1} such that $D^{L-1} \in [D^{L-1}_{min}, D^L]$. The search proceeds iteratively and only test values of delay D^l, such that $D^l \in [D^l_{min}, D^{l+1}]$, for $l = L..1$. Using this simple method, the set of playback delays that minimizes Eq. (11) can be identified with high probability for most cost functions φ that tends to minimize the average playback delay. The search space of feasible solutions is however drastically reduced compared to a full search algorithm.

D. Fair penalty distribution

In order to have a fair policy among the different clients R^l, we can distribute the playback delays such that the standard deviation of the penalties $\Delta = [\Delta^1, \ldots, \Delta^L]$ is minimized. If the average penalty is given by $\mu = E[\Delta^l]$, the playback delays can thus be chosen such that

$$\mathcal{D}_{opt} = \arg\min_{\mathcal{D}} E\left[(\Delta - \mu)^2\right] \tag{12}$$

under the condition that $S^l_{\mathcal{D}}(t) \leq C(t)\ \forall l, 1 \leq l \leq L$, i.e., all the traces are schedulable from Eq. (1).

We propose a low complexity algorithm for computing the optimal playback delay set $\mathcal{D}_{opt}$ in the sense of Eq. (12). We can observe that the minimal value of the cost function is reached when all the penalties are equivalent, i.e., $\Delta_k = \Delta_l,\ \forall k, l$. Any set of delays $\mathcal{D} = \left\{D^l | D^l = D^l_{min} + \Delta_l\right\}_{l=1}^{L}$, where $\Delta_l = K, \forall l$ minimizes the cost of Eq. (11). In other words the source traces of all layers need to be delayed by K units relative to their respective minimal playback delay D^l_{min}. Given the set of minimum playback delays $\mathcal{D}_{min}$, we can construct an aggregate source rate trace $S^L_{\mathcal{D}_{min}}(t)$, defined as:

$$S^L_{\mathcal{D}_{min}}(t) = \sum_{l=1}^{L} \Lambda^l\left(t - D^l_{min}\right). \tag{13}$$

If the trace $S^L_{\mathcal{D}_{min}}(t)$ is schedulable, it represents an ideal solution where all layers can be decoded jointly with minimal delay. If it is not the case, the playback delay can be increased in the same manner for all layers, such the trace becomes schedulable. It corresponds to shifting the aggregate source trace by the smallest delay K, such that $S^L_{\mathcal{D}_{min}}(t - K) \leq C(t), \forall t$. In other words, we can compute K as

$$K = h\left(S^L_{\mathcal{D}_{min}}, C\right), \tag{14}$$

and that can be achieved by running the Algorithm 1. Hence the complexity involved in finding the solution is that of the bisection search algorithm used in Algorithm 1, i.e., $O(log(trace\_length))$. The solution is obviously equivalent to the optimal solution of the algorithm in the previous section, when it lies in the reduced search space.

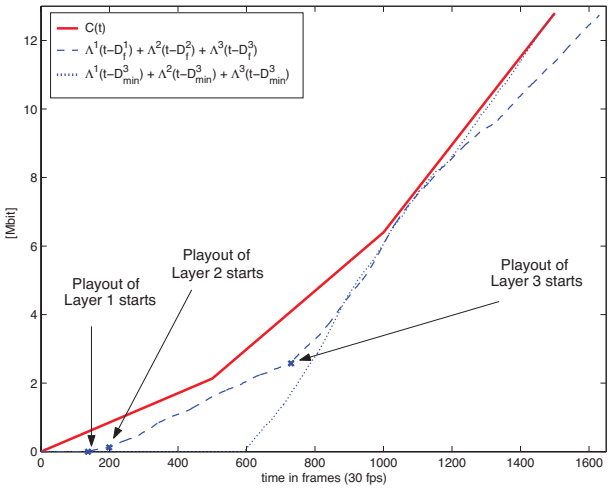

Fig. 4. The channel can support 3 layers of the encoded stream. The dashed curve shows the aggregate playout curve of the 3 layers with fair values of the playback delays $\mathcal{D}_f$. The aggregate playout curve of the 3 layers using playback delay D^3_{min} is shown for reference (dotted line).

We illustrate the solution with fair distribution of the delay penalties in Figure 4. We have encoded a composite video sequences in QCIF format at 30 frame per second, using the MoMuSys MPEG-4 FGS [16] reference codec. The channel is a piecewise CBR channel that provides a mean rate of 128kbps at the beginning, then improves to 256kbps and finally to 384kbps. Using the fair playback delay distribution proposed above, the playout can begin after a playback delay equivalent to 137 frames at receivers of set R^1. The playback delays for layer 2 and 3 are of 199 and 730 frames respectively. The relative playback delay penalty per client set, compared to their respective D^l_{min} value, is equivalent to 135 frames for all clients. Note that the gain in delay for clients in sets R^1 and R^2 is enormous when compared to a strategy that would have the same delay $D^3_{min} = 595$ frames for all clients (dotted line).

E. Unequal delay penalties

Some applications may necessitate to devise a scheduling strategy with unequal delay penalties, where some clients are considered as prioritized compared to others. This can be achieved by minimizing a weighted standard deviation of the delay penalties. In this case, the delay distribution has to be chosen according to the following optimization problem:

$$\mathcal{D}_{opt} = \arg\min_{\mathcal{D}} E\left[(w^T(\Delta - \mu)^2\right] \tag{15}$$

under the condition that $S^l_{\mathcal{D}}(t) \leq C(t)\ \forall l, 1 \leq l \leq L$, i.e., all the traces are schedulable from Eq. (1). The weights $w = [w^1, \ldots, w^L]$ represent positive weights that permit to control the distribution of the penalties among the L layers. A relatively high value of the weight w^l typically constrains the delay D^l to be close to the minimal delay D^l_{min}.

Depending on the weight distribution, it might be difficult to find the optimal solution to the problem of Eq. (15) without using an exhaustive search over the (reduced) space of possible delay values. We however propose a low complexity algorithm that finds the optimal playback delay set $\mathcal{D}_{opt}$. It is based on the a priori information about the structure of the optimal solution that sets the cost function in Eq. (15) to 0. It can be reached only when

$$\Delta^l = \frac{w^k}{w^l}\Delta^k, \tag{16}$$

$\forall k, l, 1 \leq k \leq L, 1 \leq l \leq L$. This imposes that the delay penalty takes the form $\Delta_l = K/w^l, \forall l$, where K is a constant. Therefore, solving the optimization problem of Eq. (15) is equivalent to find to smallest value of K such that the aggregate trace $S^L_{\mathcal{D}}(t)$ is schedulable. In other words, one has to find the smallest K such that verifies

$$\sum_{l=1}^{L} \Lambda^l\left(t - D^l_{min} - \frac{K}{w^l}\right) \leq C(t), \forall t. \tag{17}$$

The search algorithm, given in Algorithm 3, simply increases K gradually, until the aggregate trace is schedulable. At each iteration, it updates the playback delays, constructs the aggregate source trace, and checks the schedulability condition. If the resulting trace is schedulable, the algorithm stops. Otherwise, the value of K is augmented by δ, and the process is repeated until the resulting trace is schedulable.

Note that the algorithm may find a sub-optimal solution to the problem of Eq. (15) due to granularity of the delay increments. However, the complexity is drastically reduced compared to a full search algorithm. If all the weights are equal, we obviously get back to the the *fair* model of the previous section. However, we have seen that in the *fair* case we only need to test $O(log(trace\_length))$ possibilities, where each test can be performed in polynomial time. Algorithm 3 always performs $O(trace\_length)$ such tests.

Algorithm 3 $\mathcal{D}_h = \left(C(t), \{\Lambda^l(t)\}, w, \delta\right)$

1: $D_h^l \Leftarrow D_{min}^l$, $1 \leq l \leq L$.
2: Construct the trace using delays D_{min}^l:
3: $S_{\mathcal{D}_h}^L(t) \Leftarrow \sum_{l=1}^{L} \Lambda^l\left(t - D_{min}^l\right)$
4: **while** $S_{\mathcal{D}_h}^L(t) \not\leq C(t), \forall t$ **do**
5: increase the delays according to the assigned weights:
6: **for** $l = 1$ to L **do**
7: $D_h^l \Leftarrow D_h^l + \frac{\delta}{w^l}$
8: **end for**
9: bound delays such that there are non-decreasing:
10: **for** $l = L-1$ downto 1 **do**
11: $D_h^l \Leftarrow \left(min(D_h^l, D_h^{l+1}\right)$
12: **end for**
13: construct new trace:
14: $S_{\mathcal{D}_h}^L(t) \Leftarrow \sum_{l=1}^{L} \Lambda^l\left(t - D_h^l\right)$
15: **end while**
16: **for** $l = 1$ to L **do**
17: Playback delays are integers (in frame units)
18: $D_h^l \Leftarrow \lceil D_h^l \rceil$
19: **end for**

We validate the proposed algorithm on a composite sequence, encoded using the MoMuSys MPEG-4 FGS reference codec [17]. We run 100 tests where the channel is a piecewise CBR channel with rates chosen randomly in {128kbps, 256kbps, 384kbps}, and random lengths for each constant rate segment. We set the weights to $w = \{1, 100, 1\}$, which means that the playback delay for layer 2 in the optimal playback delay set should be kept as close as possible to its minimum playback delay. In our results, the optimal playback delay for layer 2 is indeed always within at most 2 frames of D_{min}^2, thus validating our weighted metric function, as expressed in Eq. (15). In all the cases, the cost function is minimal. Note that this might not be always the case for the algorithm with reduced search space proposed in Section III-C, since the bounds of the delay interval might not allow to find the optimal solution in the sense of Eq. (15) that does not include an explicit minimization of the average playback delay. Finally, the average number of potential solutions tested by the proposed algorithm was $1.59 \cdot 10^3$ for the aforementioned experiment, compared to $1.82 \cdot 10^7$ for the generic full search algorithm in Section III-C. The considerations on the structure of the optimal solution space thus permits a dramatic reduction of the computation time.

IV. Minimum receiver buffer

A. β-optimal sending rate

Once playback delays are given, the server still has the flexibility to choose the packet scheduling policy under the constraints given by the channel. The packet scheduling policy typically influences the dynamic behavior of the receiver buffer. In particular, we are interested in defining the sending rate at the server, which minimizes the buffer occupancy at all times t at the receiver in a given streaming scenario represented by $(C(t), S(t), D)$. At the same time, the sending rate shall ensure that the receiver buffer does not experience any starvation in order to guarantee a smooth video playback. This sending rate is called β-optimal and we denote it as $X_\beta(t)$.

If condition of Eq. (1) is verified, there exists a family of sending rates $\mathcal{X}$ such that each $X(t) \in \mathcal{X}$ satisfies both Eqs. (2) and (3). In these cases, the video can be played back at the receiver after D time units without experiencing any buffer underflow. The β-optimal sending rate is the scheduling solution that minimizes the buffer occupancy at the receiver. It can be written as:

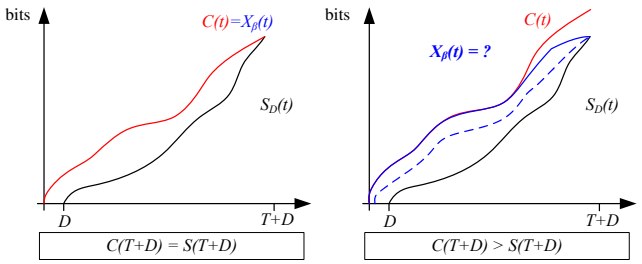

Fig. 5. *Left:* Limiting case with $X_\beta(t) = C(t)$. *Right:* The set of sending traces $X(t)$ that verifies Eqs. (2) and (3) generally contains multiple candidates.

$$X_\beta(t) \quad = \quad \arg \min_{X(t) \in \mathcal{X}} \left(B(t) = X(t) - S_D(t)\right), \forall t, \qquad (18)$$

which means that, for any sending rate $X(t) \in \mathcal{X} \setminus X_\beta(t)$, we have:

$$X_\beta(t) \leq X(t), \forall t \qquad (19)$$

Using the formalism from [4], $X_\beta(t)$ is the smallest sending rate that satisfies the following conditions:

- $X(t)$ is a causal flow, i.e., $X(t) = 0$ for $t \leq 0$.
- the flow $X(t)$ is constrained by an arrival curve $\sigma(\cdot)$ that reflects the channel availability constraints. This means that for all $t \geq 0$ and for all $s \in [0, t]$, $X(t) - X(s) \leq \sigma(t - s)$. There is no further constraint imposed by the network.

When the server can prefetch data from any future frame at any time, and when the playback delay is chosen such that there is no buffer underflow at the receiver, there exists a minimal solution to the above set of constraints. It is given by:

$$X_\beta(t) \quad = \quad (S \oslash \sigma)(t - D) \qquad (20)$$

Here $\oslash$ denotes the *Min-Plus deconvolution* of two wide-sense increasing functions f and g, defined as:

$$(f \oslash g)(t) \quad = \quad \sup_{u: u \geq 0} \{f(t + u) - g(u)\} \qquad (21)$$

The interested reader is referred to [5] for more details on the network calculus formalism that is used for proving the existence of the minimal sending trace.

In the rest of this section, we propose an algorithm that offers an intuitive and tractable solution for computing the β-optimal sending rate for non-scalable streams. This algorithm is a generalization of the algorithm presented in [1]. We then show that a *jointly* β-optimal sending trace exists also in the case of scalable streams, and we propose a method to compute the sending rate that minimizes the buffer occupancy for a set of heterogeneous receivers.

B. Single layer streams

We provide an intuitive algorithm for computing the β-optimal sending rate for single layer streams. Let us consider first a limiting case where the channel has to be fully used to transmit the complete bitstream, i.e., $C(D + T) = S_D(D + T)$. In this case illustrated in Figure 5 (left), the set of schedulable sending traces only contains one solution. Any sending rate for which there exists some t where $X(t) < C(t)$ implies that $X(D + T) < S_D(D + T)$, where T is the duration of the video sequence. This violates the condition of Eq. (2). Hence the only valid sending rate function is also the solution that minimizes the buffer occupancy for all times t. It is given by $X_\beta(t) = C(t)$.

Algorithm 4 $X_\beta = VRS\left(C(t), S_D(t)\right))$

Require: $S_D(t) \leq C(t), \forall t$

$X_\beta(t) \Leftarrow C(t)$, for all t. The sending trace is computed as a reduced channel trace.
$t \Leftarrow 0$
while $t \leq T + D$ **do**
  if $\nexists t' \geq t$ s.t. $X_\beta(t') = S_D(t')$ **then**
    Reduce the channel down by $X_\beta(t) - S_D(t))$ bits.
    for all τ in $[t, T + D]$ **do**
      $X_\beta(\tau) \Leftarrow X_\beta(\tau) - X_\beta(t) + S_D(t)$
    end for
    $t \Leftarrow t + 1$
  else
    The curves touch, no reduction at this step.
    $t_{new} \Leftarrow \sup_{\tau > t} \{\tau | X_\beta(\tau) = S_D(\tau)\}$
    $t \Leftarrow t_{new}$
  end if
end while

In the general case where $C(T+D) > S_D(T+D)$, several sending traces represent valid scheduling solutions that satisfy the condition of Eq. (1), as illustrated in Figure 5 (right). In order to compute the β-optimal sending rate, we make the following observations. First, $X_\beta(t)$ obviously shall fulfill the conditions of Eqs (2) and (3) that define the schedulable solutions. In order to minimize the buffer occupancy $B(t)$, $\forall t$, $X_\beta(t)$ also needs to follow $S_D(t)$ as closely as possible. This is equivalent to keeping the sending rate as small as possible, but still to send enough data to avoid buffer starvation under the constraints imposed by the channel bandwidth. Finally, we know from the limiting case presented above that, whenever there exists a time τ such that $S_D(\tau) = C(\tau)$, the β-optimal sending rate needs to be equal to $C(t)$ up to τ. Therefore, it becomes clear that $B(t)$ can be minimized for all times t if and only if data is sent at the latest possible instant in time such that all data still arrive on time for decoding. We can thus eliminate the early sending opportunities offered by the transmission channel, and *reduce* the channel to the sending opportunities that are *necessary* to transmit all the data before their decoding timestamps.

The β-optimal sending rate can now be computed by the *Variable Rate Smoothing* (VRS) algorithm given in Algorithm 4. The algorithm operates in the cumulative domain, starting at time $t = 0$. It first sets the sending trace to be equal to the channel trace $C(t)$, $\forall t$. Then, it iteratively checks for $t = 0 \ldots T + D$ whether there is equality at any future time instant t' between the sending trace and the delayed source trace $S_D(t)$. In this case, the situation is similar to the limiting case presented above, and the sending rate has to be equivalent to the channel rate up to the time instant $t' = t$. However, if the sending trace is strictly larger than the delayed source trace for all time instants $t' > t$, the sending trace at all time instants $t' > t$ is reduced by the difference between the sending trace and the delayed source trace at time t. This operation basically consists in eliminating the early transmission opportunities, which would results in wasting buffer resources. It is equivalent to translating the sending trace curve down by $X_\beta(t) - S_D(t)$ for all $t' > t$. Note that the complexity of Algorithm 4 is $O(T + D)$, where the worst case is achieved if $s(t - D) < c(t)$, $\forall t$.

An illustration of the VRS algorithm is presented in Figure 6, where the channel trace $C(t)$ is linear and the delayed source trace $S_D(t)$ is piecewise linear. At time $t = 0$, $X_\beta(t)$ and $S_D(t)$ do not touch. This will remain the same up to t^1. This means that up to t^1, the derivative of $S_D(t)$ is certainly never larger than that of X_β, or equivalently that the instantaneous sending rate is not smaller than the delayed source rate. The algorithm sets the sending trace to $S_D(t)$ up to time t^1. The sending trace computed at time $t = t^1$ touches the source curve at time t^2_{new}. This means that, in the interval $[t^1, t^2_{new}]$, the derivative of $S_D(t)$ is at times larger than the derivative of the sending trace of $X_\beta(t)$. In other words we have reduced the situation in this interval to the limiting case and we have to use all the channel bits in order to transmit the needed data. The algorithm does not reduce the sending trace for t in $[t^1, t^2_{new}]$. After $t = t^2_{new}$, the sending trace can again be reduced to the delayed source trace. Using the time-inversion technique outlined in [5], it can finally be checked that the resulting sending rate in this example is the same as the one resulting from Eq. (20) where the arrival curve is set to $\sigma(t) = \frac{dC(t)}{dt} \cdot t$.

Once the optimal sending trace has been computed, the buffer optimal scheduling strategy simply consists in sending data in the increasing order of their decoding deadline while respecting the constraints given by the sending trace. If one does not respect this order, some packets are sent in advance, which can only contribute to increase the buffer occupancy. Equivalently, the optimal scheduling of the packets can also be achieved by scheduling packets as late as possible [9], without pre-computing the buffer optimal sending trace. This last opportunity scheduling policy basically consists in reversing time, starting from $t = T+D$ to $t = 0$. Then it schedules at each time instant t as many of the packets with the largest decoding deadlines in $s_D(t)$ as the channel $c(t)$ permits it. This solution jointly computes the buffer optimal scheduling, and the optimal sending trace. It is however based on reversing the time axis after $t = T + D$, which might present limitations in some practical systems.

Finally, Figure 7 illustrates the β-optimal sending rate: the used video trace is formed of 2 GOPs of the MPEG-4 encoded Foreman sequence. The channel is a constant bit rate (CBR) channel. The left column depicts the source rate $s(t)$, channel rate $c(t)$ and the illustrated sending rate. The middle column shows the same traces in the cumulative domain, and the right column shows the buffer occupancy as a function of time: $B(t) = X(t) - S_D(t)$. The top row shows one valid but sub-optimal sending trace. Note that $\sup_t \left(X(t) - S_D(t)\right) = 21264 > 11468$ bits (see *top-right*). The bottom row shows the β-optimal scheduling policy, where the sending rate follows the source rate whenever possible. Whenever $s_D(t) > c(t)$, data is sent at the latest possible opportunity, thus minimizing the buffer occupancy for all t. The maximum amount of buffering needed is 11468 bits (see *bottom-right*).

C. Scalable streams

In this section, we consider the case of scalable streams and we show that there exists a scheduling strategy that jointly minimizes the buffer occupancy $B^l(t)$ for each each receiver group R^l and at all times t. Then we propose a scheduling algorithm that offers a practical solution to build the sending trace $X_\beta(t)$ that is jointly β-optimal for multiple receivers.

We consider that a set of source traces $\mathcal{S} = \left\{\Lambda^l(t)\right\}_{l=1}^{L}$, representing L additive hierarchically encoded layers are sent simultaneously to multiple receivers $R^l, 1 \leq l \leq L$ through a joint bottleneck channel given by the cumulative rate $C(t)$. We further consider a set of non-decreasing playback delays $\mathcal{D} = \left\{D^l\right\}_{l=1}^{L}$ that are used by the different receiver groups. Each receiver in the group R^l starts consuming the media layers 1 to l of the hierarchically encoded stream after an initial playback delay D^l. The *aggregate source trace* that has to be sent over the channel in order to ensure a smooth

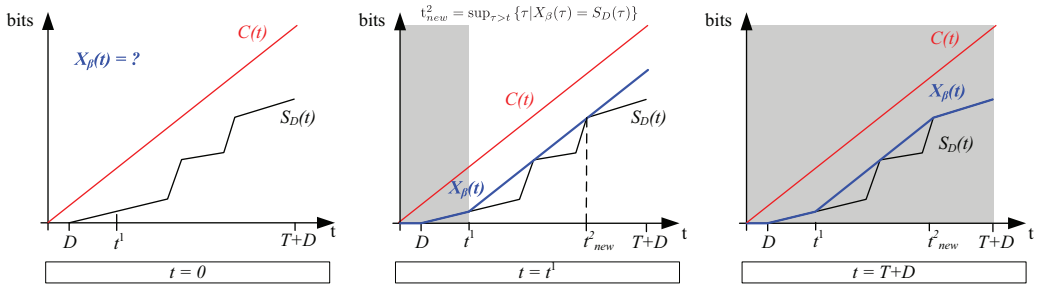

Fig. 6. Illustration of the VRS Algorithm. In order to minimize the buffer occupancy, the sending trace is reduced to the delayed source trace when the channel rate is superior to the source rate.

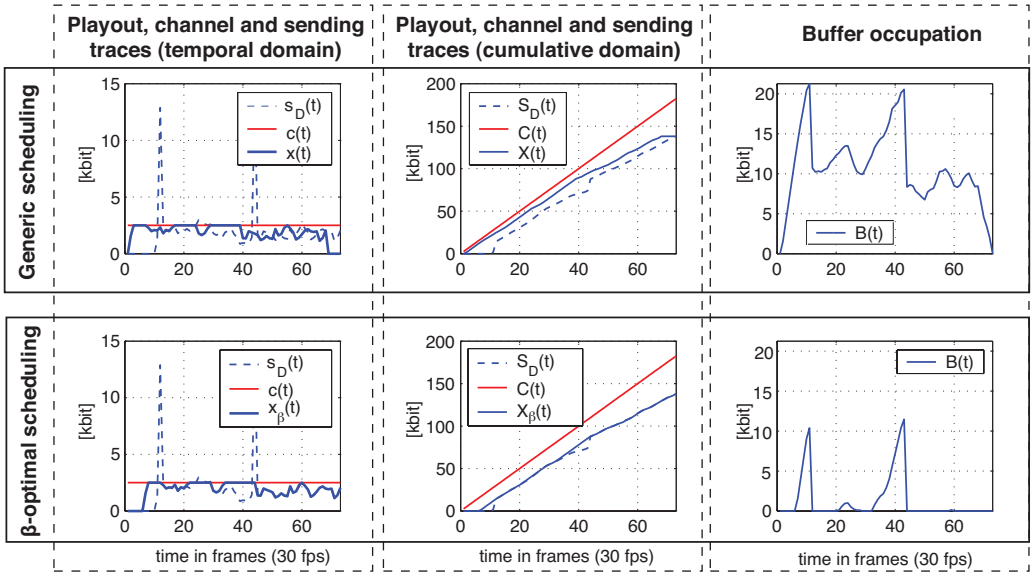

Fig. 7. β-optimal scheduling outperforms any generic scheduling algorithm with respect to receiver buffer occupancy.

playback by all decoders is constructed as:

$$S^l_{\mathcal{D}}(t) \quad = \quad \sum_{i=1}^{l} \Lambda^i_{D^i}(t), \tag{22}$$

where $\Lambda^i_{D^i}(t)$ is the cumulative function of $\lambda^i_{D^i}(t)$, the source trace of layer i, delayed by D^i. We assume here that $\mathcal{D}$ is chosen such that the full stream is schedulable, i.e. that $S^l_{\mathcal{D}}(t) \leq C(t)$, $\forall t$, $\forall l$. It is important to note here that the cumulative rate given by Eq. (22) is in fact larger than the rate used by the decoder. When a receiver in R^l starts playing the stream at time D^l, all the source traces up to l are drained simultaneously from the receiver buffer. Thus the *playout trace* at a receiver in R^l, which represents the number of bits consumed up to time t, is given as:

$$S^l_{D^l}(t) \quad = \quad \sum_{i=1}^{l} \Lambda^i_{D^l}(t). \tag{23}$$

These different traces are illustrated in Figure 8. Note that we have:

$$X^l(t) \geq S^l_{\mathcal{D}}(t) \quad \geq \quad S^l_{D^l}(t), \forall t, \tag{24}$$

where the first inequality is due to the schedulability condition, and the second inequality results from the construction of the playout trace, with $D^i \leq D^{i+1}$. There are in general several valid sending traces $X^l(t)$ for scheduling the layers 1 to l under a given set of delay and source trace constraints. We denote this set of valid traces as $\mathcal{X}^l = \{X^l(t)\}$. We are interested in finding the trace $X^l_\beta(t) \in \mathcal{X}^l$ that minimizes the buffer occupancy all the receivers R^l for all times t. It corresponds to the sending trace that minimizes $B^l(t) = X^l(t) - S^l_{\mathcal{D}}(t)$, $\forall t$. The cumulative sending trace is built on l additive layers, and we denote the sending trace of layer l as $Y^l(t)$, with $\sum_{k=1}^{l} Y^k(t) = X^l(t)$. Similarly, we denote the β-optimal sending trace of layer l as $Y^l_\beta(t)$.

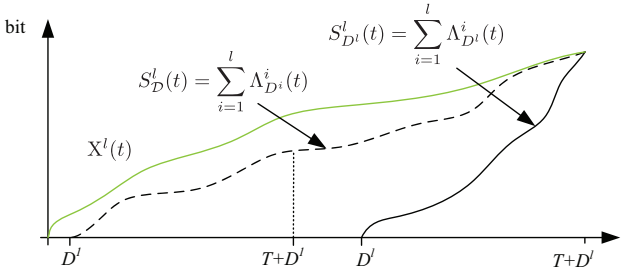

Fig. 8. Illustration of the aggregate source trace $S^l_{\mathcal{D}}(t)$ and the play-out trace at receiver R^l, $S^l_{\mathcal{D}}(t)$, along with one of the possible sending traces $X^l(t)$.

From the previous section, we know that if we only consider one resolution level l, $X^l_\beta(t)$ exists. It can be computed by Algorithm 4 for every resolution level l. In a scenario where clients might subscribe only to a subpart of this aggregated stream for a resolution level $k < l$, such a scheduling would however be suboptimal in terms

of buffer occupancy for the low resolution clients. In other words, if the sending rate is generated from the stream at resolution l without explicitly considering the lower layers, we can a priori not provide any guarantee on the buffer occupancy at receivers R^k, $k < l$ that consume only the lower layers. We are rather interested in finding the sending trace that minimizes the buffer occupancy at all times t for all receivers simultaneously, if such a solution exists. We prove below an important proposition that says that a joint β-optimal scheduling for multiple receivers actually exists, and that the solution $X^l_\beta(t) \in \mathcal{X}^l$ can actually be constructed on the β-optimal traces for layers $k < l$.

Proposition 1: If $S^L_{\mathcal{D}}(t) \leq C(t), \forall t$, then there exists a scheduling policy that is jointly β-optimal for all receivers R^l, $1 \leq l \leq L$ that respectively consume the layers 1 to l after an initial playback delay D^l.

Proof:

Given $C(t)$ and $S^l_{\mathcal{D}}(t)$, we know that $X^l_\beta(t)$ exists, for any l taken individually. We want to show that there exists a valid sending trace $Y^l_\beta(t)$ for scheduling layer l, when streams at resolution l and $l-1$ both minimize the buffer occupancy for the respective client sets. Such a trace can be written as:

$$Y^l_\beta(t) = X^l_\beta(t) - X^{l-1}_\beta(t). \tag{25}$$

It is a valid sending trace for layer l iff

$$\Lambda^l_{D^l}(t) \leq Y^l_\beta(t) \leq C(t) - X^{l-1}_\beta(t), \forall t. \tag{26}$$

In other words, the sending trace for layer l has to be large enough to ensure a smooth playback after a delay D^l. At the same time, it has to be small enough to respect the channel constraints, once the sending trace $X^{l-1}_\beta(t)$ has been allocated already. As $X^l_\beta(t)$ is schedulable by hypothesis, we have $X^l_\beta(t) \leq C(t)$. We can therefore write $X^l_\beta(t) - X^{l-1}_\beta(t) \leq C(t) - X^{l-1}_\beta(t)$ Combined with Eq. (25), it leads to proving the second part of Eq. (26).

The schedulability of $X^l_\beta(t)$ also induces that the data of layer l are present on time at the decoder. In other words, there exists a set of sending traces $X^{l-1}(t)$ for the data of layers 1 to $l-1$ such that

$$\Lambda^l_{D^l}(t) \leq X^l_\beta(t) - X^{l-1}(t), \forall t.$$

In particular, since by definition $X^{l-1}_\beta(t) \leq X^{l-1}(t)$, we have

$$\Lambda^l_{D^l}(t) \leq X^l_\beta(t) - X^{l-1}_\beta(t), \forall t,$$

or equivalently

$$\Lambda^l_{D^l}(t) \leq Y^l_\beta(t),$$

which proves the first part of Eq. (26). ■

Therefore, there exists a valid sending trace that minimizes jointly the buffer occupancy for receivers sets R^{l-1} and R^l. By recursion, we can construct the β-optimal solutions as $X^l_\beta(t) = \sum_{k=1}^{l} Y^k_\beta(t)$. Moreover, we have the following corollary.

Corollary 1: The optimal sending trace $Y^l_\beta(t)$ is the minimal valid sending trace for layer l on a channel of bandwidth $C(t) - X^{l-1}_\beta(t)$, when the playback delay is set to D^l.

Proof: We prove the corollary by contradiction. Assume that there exists a trace $Y_l(t)$ such that $Y_l(t) < Y^l_\beta(t)$ as some time t. In this case, we have $X^l(t) = X^{l-1}_\beta(t) + Y^l(t) \leq X^l_\beta(t)$, which contradicts the assumption on the optimality of $X^l_\beta(t)$. $Y^l_\beta(t)$ is therefore the minimal sending trace for layer l. ■

Based on Proposition 1 and Corollary 1, we can build an iterative algorithm to build the joint β-optimal sending rate for any layer l by greedily building the β-optimal sending rate one layer at a time, starting at the lowest one. In particular, we have

$$X^l_\beta(t) = \sum_{i=1}^{l} Y^i_\beta(t), l > 1 \tag{27}$$

and $Y^1_\beta(t) = X^1_\beta(t)$. The sending rate can be computed for each layer iteratively starting from layer 1 by the VRS algorithm. It computes the sending trace that corresponds to $\Lambda^l_{D^l}$, the bits of layer l that are decoded after a playback delay D^l. The bandwidth constraints are updated iteratively, as the bandwidth used by the lower layers is removed from the channel capacity. Therefore, we have

$$C^l(t) = C^{l-1}(t) - Y^{l-1}_\beta(t), \forall 1 < l \leq L, \tag{28}$$

where $C^l(t)$ corresponds to the part of the channel that is available to schedule bits from layer l, and $C^1 = C(t)$. Once the optimal sending traces $Y^i_\beta(t)$ have been computed for each layer l, the packet scheduler proceeds by sending the data of each layer in the increasing order of the decoding deadlines, while respecting the different sending traces. For each layer, the scheduler proceeds similarly to the scheduler for single layer streams.

Note that another strategy could be proposed to reach the buffer optimal sending traces. It consists in reverting the time axis, starting from $t = T + D^l$ to $t = 0$. Then the packets of each layer are sent at the latest moment for correct decoding, while respecting the channel bandwidth $c(t)$. As the layer 1 is decoded with the smallest playback delay, it is scheduled first. Other layers are scheduled iteratively under the constraints given by the remaining channel bandwidth $c^l(t)$. Similarly to the case of single layer streams, this solution guarantees the lowest buffer occupancy at the decoder, without the explicit computation of the optimal sending traces. From Proposition 1, it also leads to the jointly optimal policy for all the resolution levels, or all the clients R^l.

We illustrate the resulting performance of this iterative algorithm in Figure 9: it shows a constant bitrate channel and the playout rates of 2 GOPs of the MPEG-4 FGS encoded Formeman sequence, at receiver R^1 (layer 1 only) and R^2 (layers 1 and 2), see *left*. Playout begins at all the receivers after D=20 frames. The same scenario is shown in the cumulative domain (see *middle*). Only the aggregate playout trace at R^2 (i.e. $\Lambda^1(t-D)+\Lambda^2(t-D)$) is shown in blue. The green curve shows the β-optimal sending rate for the aggregate playout curve and the considered channel, as given by the VRS Algorithm. It is thus the β-optimal sending rate for receivers in the set R^2. Figure 9-*right*: on the one hand, the solid and dashed blue curves show the sending rates for layer 1 data and layer 2 data respectively, in the case where the β-optimal sending trace is computed from the aggregate playout trace at R^2. On the other hand, the dashed red curve shows the β-optimal sending rate for R^1, $X^1_\beta(t)$, obtained from a β-optimal scheduling of layer 1 over the channel $C(t)$. It can be noticed that data for layer 1 is only transmitted shortly before the playout deadline, thus reducing the buffer occupancy at R^1. In this scenario, the solid red curve shows the sending rate of layer 2 over the remaining bitrate $C(t) - X^1_\beta(t)$. Note that in both cases, all data that is sent meets their deadline, and in both cases, the two respective sending rates add up to $X_\beta(t)$ (green curve), which is the β-optimal sending rate for R^2.

In the same scenario, Figure 10 finally shows the evolution of the buffer occupancy at receivers R^1 (left) and R^2 (right) if joint β-optimal scheduling is used (top) and if β-optimal scheduling is computed only on the 2 layers stream (bottom). The minimum buffer occupancy at R^2 is achieved in both cases, however the minimum buffer occupancy at R^1 is only achieved in the first case (top-left). The joint β-optimality is achieved through the fact that receivers that subscribe to higher layers buffer less data from lower layers in the first case, and more data from higher layers. However, the buffer contains the same total amount of data in both scheduling choices.

Finally, it is important to note that joint playback delay and buffer optimization can be achieved with the algorithms proposed in the Sections III and IV. The delay optimization does not put assumptions on

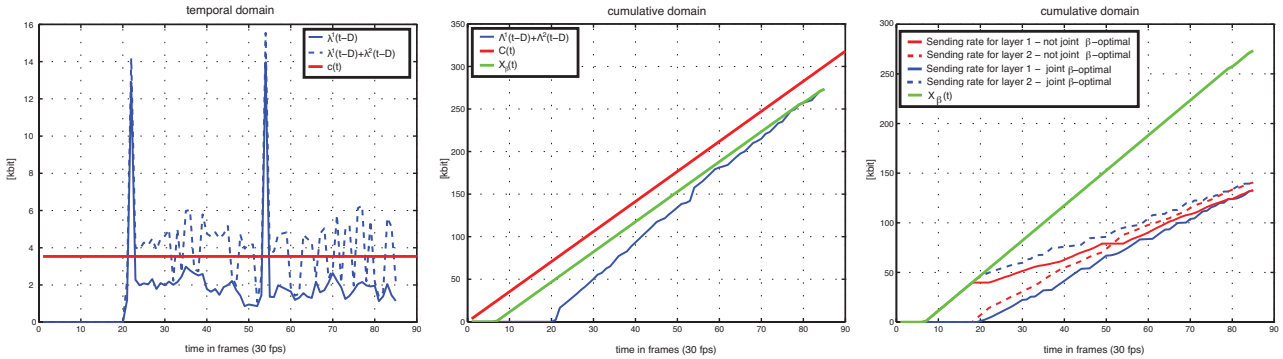

Fig. 9. Validation of the β-optimal scheduling scheduling algorithm. *Left:* the traces of two layers and a CBR channel in the temporal domain, playout starts after 20 frames. *Middle:* The channel trace and the aggregate playout trace for both layers in the cumulative domain. The green curve shows the sending trace the minimizes the buffer occupancy at R^2, as given by the VRS algorithm. *Right:* We achieve the β-optimal sending trace for layer 1, without sacrificing the β-optimality of the aggregate sending trace for both layers 1 and 2.

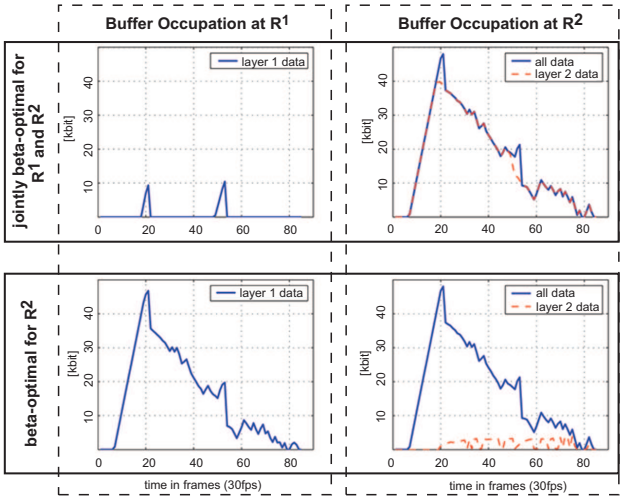

Fig. 10. Buffer evolution in β-optimal scheduling scenarios.

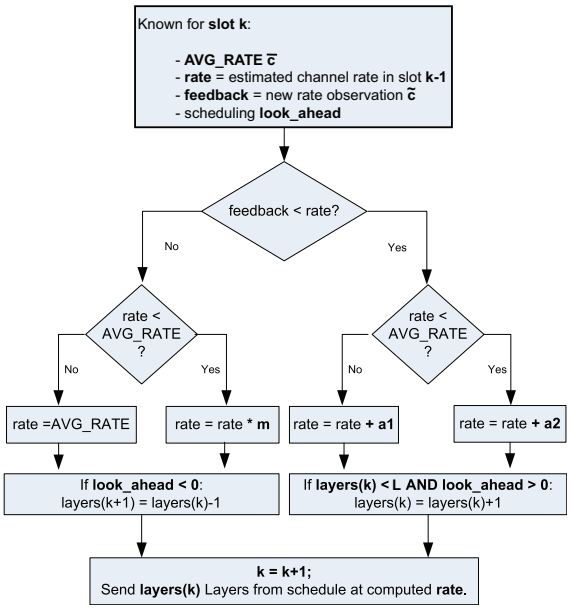

Fig. 11. Rate adaptation algorithm

the actual sending traces,it only considers schedulability conditions. Similarly, when playback delays are selected, the buffer optimization simply consists in finding the smallest sending trace among the set of valid traces. Both problems can be solved sequentially, and the resulting solution jointly optimizes the playback delay, and the buffer occupancy.

V. Channel-adaptive streaming

A. Source rate adaptation

In the previous sections, we have provided an analysis of the playback delay and buffer occupancy, as well as joint optimization strategies. These solutions rely on the assumption of perfect knowledge about the bottleneck channel bandwidth. They provide upper-bounds on the performance of common practical systems, where the complete channel trace is usually not known at the server. When the actual channel bandwidth does not exactly correspond to the trace that is used for packet scheduling, the server may not be able to send all packets according to their computed schedules. It has therefore to take actions such as reduction in the source rate, to adapt to temporary bandwidth reduction. Rate adaptation can be performed efficiently on scalable streams by dropping packets from the higher layers. If such mechanisms are used carefully, the quality of service is not significantly affected. Another solution is to devise a conservative scheduling approach that considers lower-bounds on the channel bandwidth. The authors in [18] for example compute the playback delay for a single stream over a *stochastic* channel by deriving a channel trace that lower bounds all possible realizations of the channel. Rate adaptation generally reaches a higher average quality than conservative scheduling methods, at the price of possibly higher quality variations for the clients that subscribe to the highest resolution streams.

We assume that the server knows some channel statistics such as the average bottleneck bandwidth $\bar{c}$. The playback delays and the sending traces are initially computed based on a constant bit rate channel of rate $\bar{c}$. This results in a complete schedule that determines which packet (and which layer) has to be sent at each time instant t, in an ideal scenario. During the broadcasting session, the server monitors the state of the channel, and the sending rate can be adapted in case the available bandwidth becomes insufficient to be able to respect the original packet schedule.

We propose below a sample system based on a simple rate adaptation algorithm, and we show that rate adaptation still permits to keep the buffer occupation close to minimum, and that playback

delays close to the ideal values can be achieved, at the price of only minor and controlled PSNR degradations.

B. System description

We have tested the rate adaptation scheme on a sample system. The scalable video stream is segmented such that data from different frames and different video layers are fed into different RTP packets. The video stream is sent simultaneously to 3 clients R^1, R^2 and R^3 that decode layers up to 1, 2 and 3 respectively. The average channel rate has been set to 32 kbytes/sec, and we use a NISTNet [19] network emulator to limit the bandwidth on the server-client broadcast link according to a given random bandwidth trace, which is unknown at the server.

The server sends the stored layered stream according to the scheduling strategy computed with a CBR channel of $\bar{c} = 32$ kBps and a given set of target playback delays $\mathcal{D}$. At each discrete time t the server transmits RTP packets according to the ideal scheduling plan, when it is possible. At the same time, the server updates the channel rate estimate as well as the scheduling look-ahead after each second. The approximate channel rate is computed from the round-trip time estimated that are sent in the client RTCP receiver reports, as $\tilde{c} = \frac{packet\_length*2}{RTT}$, where `packet_length` is the average length of packets that have been sent during the previous slot. The scheduling look-ahead represents the difference between the actual scheduling, and the scheduling that has been pre-computed with an ideal channel. In other words, it measures the advance that the scheduler has taken compared to the pre-computed schedule.

Based on these parameters, the rate adaptation algorithm presented in Figure 11 adapts the sending rate according to a AIMD (additive increase multiplicative decrease) policy. The additive step size is dependent on whether the current rate is above or below the targeted average rate, and we have chosen the following factors:

- $a_1 = \frac{\bar{c}}{framerate}$, if $\tilde{c} > \bar{c}$
- $a_2 = 2 \cdot \frac{\bar{c}}{framerate}$, if $\tilde{c} \leq \bar{c}$.

The choice of having a lower increment if the estimated rate is above average, leads the server to taking advantage carefully of the available rate, while trying to avoid over-estimation. The order of packets is maintained even if the rate has to be adapted. The sending rate is thus augmented by advancing faster on the pre-defined schedule, and resulting in a positive scheduling look-ahead value.

When the sending rate has to be reduced, we use a multiplicative decrease policy with a factor $m = 0.96$. However, if a decrease occurs when the rate is above the average rate, the rate is immediately clipped to the average rate. The choice of these parameters is based on empirical data and depends on the channel statistics. If the transmission is ahead of schedule, the look-ahead is used to absorb the temporary rate decay and the sending rate is simply reduced, while the order of the packets is maintained. If however, the transmission is running behind the original schedule, packets of the highest layer are not transmitted and simply dropped.

C. Experimental results

The performance of the above sample system are now analyzed through experiments. We have used the same composite video sequence as proposed before, that has been encoded in QCIF at 30 frame per second with the MPEG-4 FGS encoder. The set of target delays was set to $\mathcal{D} = \{D_1 = D_2 = 98, D_3 = 381\}$.

Figure 12 shows the number of layers that are transmitted by the rate-adaptive server, as well as the evolution of the scheduling look-ahead as compared to the pre-computed schedule. We see that at time instant $t = 23$ sec, the channel estimate is low and the scheduling look-ahead is not sufficient to continue sending all layers. So the server stops transmitting layer 3 in order to avoid further congestion until both the channel rate and the scheduling look-ahead increase again. Finally Figure 13 shows the decodable received source trace at client R^3 that subscribes to the complete stream, and starts decoding after a playback delay D_3, equivalent to 381 frames. It can be seen that approximately 3 seconds worth of layer 3 data are missing. This correspond to the the amount of layer 3 data that was not transmitted due to the server's rate adaptation. Dropping the highest layer temporarily from the broadcast leads to a decrease of less than 0.5dB in average PSNR compared to the complete reception of layer 3. However, if a conservative scheduling approach is chosen in such a scenario, the layer 3 is not transmitted at all. The average quality is therefore higher with the rate adaptation solution, at the price of quality variations.

We analyze in Figure 14 the influence of the rate adaptation on the playback delays that are necessary to ensure smooth decoding at the receivers. The target playback delays that are pre-computed in an ideal streaming scenario are $\mathcal{D} = \{D_1 = D_2 = 98, D_3 = 381\}$. These delays are obviously conservative, since they can be achieved

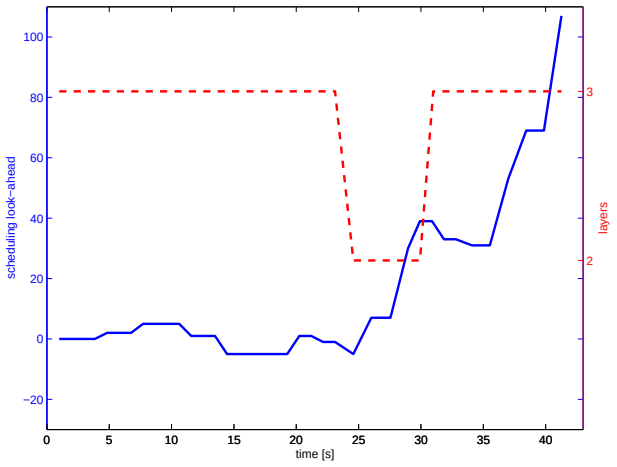

Fig. 12. Scheduling look-ahead compared to the pre-computed schedule (solid line) and number of transmitted layers (dashed line) according to the rate adaptation algorithm.

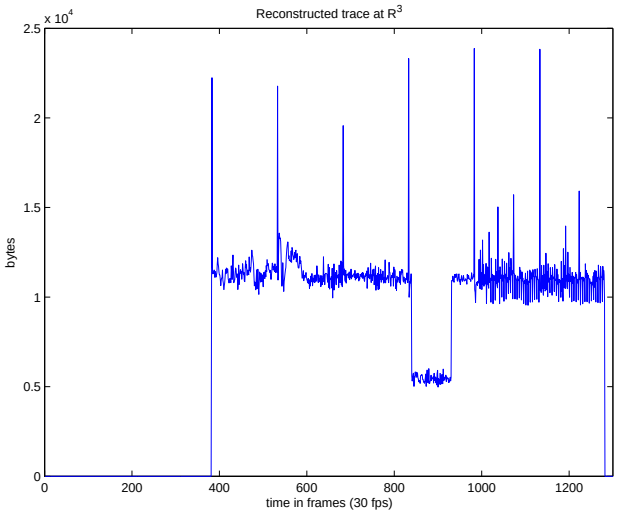

Fig. 13. Playback trace at R^3 expects to decode the complete stream (3 layers).

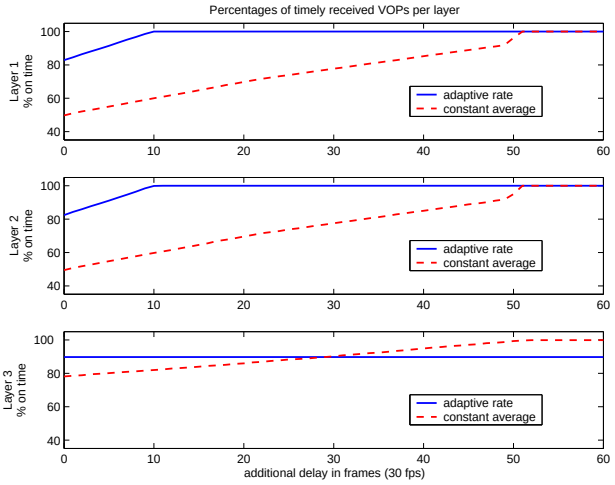

Fig. 14. Percentage of packets that are received on time as a function of a playback delay margin K. Due to the rate adaption mechansim, roughly 10% of layer 3 data are not transmitted and thus never received.

only when the channel rate exactly correspond to the previsions. In order to illustrate the influence of the rate adaptation, we represent the number of packets that arrive on time, as a function of an additional delay K used for decoding the streams (i.e., the actual playback delays are $\mathcal{D}+K$). The solid and dashed lines respectively represent the behavior of the rate adaptive scheme, and of an algorithm that does not try to adapt to the actual bandwidth and simply transmits packets according to the pre-computed schedule. It can be seen that the rate adaptive server clearly achieves better performances by keeping the necessary playback delays close to the targeted ones. If an additional delay of only 10 frames is used at the decoder, all layers can be decoded without buffer underflow. Rate adaptation therefore permits to efficiently control the quality of the transmission and to respect the timing constraints of the streaming application. A small conservative margin on the playback delays is sufficient to guarantee a smooth playback.

Finally, we analyze the buffer occupancy at the three receivers. Figure 15 illustrates the buffer fullness for playback delays of $\mathcal{D}$ and $\mathcal{D}+10$. In the second case, which ensures a smooth playback delay, we have computed the maximum difference between the actual buffer occupation and the optimal buffer occupation in the ideal scenario with a CBR channel rate of 32kbps. We can see that the difference with the ideal scenario is always lower than 10kbytes, which is a negligible penalty.

D. Discussion

The experimental results show that even a simple rate adaption algorithm based on partial channel knowledge can yield results that are close to optimal in terms of both targeted playback delays and buffer occupancy, at the expense of some minor and controllable PSNR degradations. It is worth to be noted that our experimental setup behaves like an overly *nice* network, as any injection of data at a higher rate than the actual channel rate results in a pure delay at the receiver. There are no losses due to buffer overflows (congestions) in the network. If such losses happen, we expect that the rate adaptive system is less affected than the non-adaptive server, since it makes effort to avoid congestions by changing the sending rate according to the available bandwidth.

Finally, the additional playback delay that is needed to compensate the discrepancies between estimated rate and actual channel rate can be negotiated between the server and the clients at the beginning of the streaming session. They represent a trade-off between resiliency to channel variations and the waiting time before decoding that is usually kept minimal.

VI. Conclusions

This paper has described the problem of scalable media scheduling in broadcast scenarios. In particular, we have shown the playback delay can generally not be jointly minimized for all the receivers. It typically represents the price to pay for applications where different users simultaneously subscribe to different quality levels of the same stream. We have presented a reduced complexity solution for optimizing the delay in a set of receivers. When the optimal strategy consists in minimizing the variance of the delay penalties, we have proposed low complexity algorithms that compute the optimal delay set. When delays are fixed, we have shown that there is a unique scheduling solution that minimizes the buffer occupancy at all the receivers simultaneously. If both problems are solved sequentially, one can achieve jointly an optimal delay selection and a minimal buffer occupancy. Finally, we have proposed a rate adaptation algorithm, which deals with unpredictable channel bandwidth variations. This simple scheme permits to achieve close to optimal results, even when the knowledge about the channel status is limited. It provides a viable alternative to conservative packet scheduling in practical streaming scenarios.

References

[1] J. Rexford and D. Towsley, "Smoothing variable-bit-rate video in an internetwork," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 202–215, April 1999.

[2] J. Zhang and J. Hui, "Applying traffic smoothing techniques for quality of service control in vbr video transmissions," *Computer Communications*, vol. 21, pp. 375–389, April 1998.

[3] J. McManus and K. Ross, "Video-on-demand over atm: constant-rate transmission and transport," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1087–1098, Aug 1996.

[4] P. Thiran, J.-Y. Le Boudec, and F. Worm, "Network calculus applied to optimal multimedia smoothing," in *Proceedings of Infocom*, vol. 3, pp. 1474–1483, IEEE, April 2001.

[5] J.-Y. Le Boudec and O. Verscheure, "Optimal smoothing for guaranteed service," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 689–696, Dec 2000.

[6] D. Saparilla and K. W. Ross, "Optimal Streaming of Layered Video," in *Proceedings of Infocom*, vol. 2, (Tel Aviv, Israel), pp. 737–746, IEEE, March 2000.

[7] P. de Cuetos and K. W. Ross, "Optimal streaming of layered video: joint scheduling and error concealment," in *Proceedings of ACM Multimedia*, (New York, NY, USA), pp. 55–64, ACM Press, 2003.

[8] Miao Z. and Ortega A., "Optimal scheduling for streaming of scalable media," in *Proceedings of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1357–1362, October 2000.

[9] W. Feng, "Rate-constrained bandwidth smoothing for the delivery of stored video," in *Proceedings of IS&T/SPIE Multimedia Networking and Computing*, pp. 316–327, SPIE, Feb 1997.

[10] W. Zhao and S. K. Tripathi, "Bandwidth-Efficient Continuous Media Streaming Through Optimal Multiplexing," in *Proceedings of SIGMETRICS*, (Atlanta, GA, USA), pp. 13–22, ACM, 1999.

[11] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-Driven Layered Multicast," in *Proceedings of ACM SIGCOMM*, vol. 26, (New York, NY), pp. 117–130, ACM, August 1996.

[12] Q. Zhang, Q. Guo, W. Zhu, and Y.-Q. Zhang, "Sender-Adaptive and Receiver-Driven Layered Multicast for Scalable Video Over the Internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, pp. 482–495, April 2005.

[13] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, vol. 8, April 2006.

[14] P. D. Cuetos and K. W. Ross, "Unified Framework for Optimal Video Streaming," in *Proceedings of Infocom*, vol. 3, (Hong Kong), pp. 1479–1489, IEEE, March 2004.

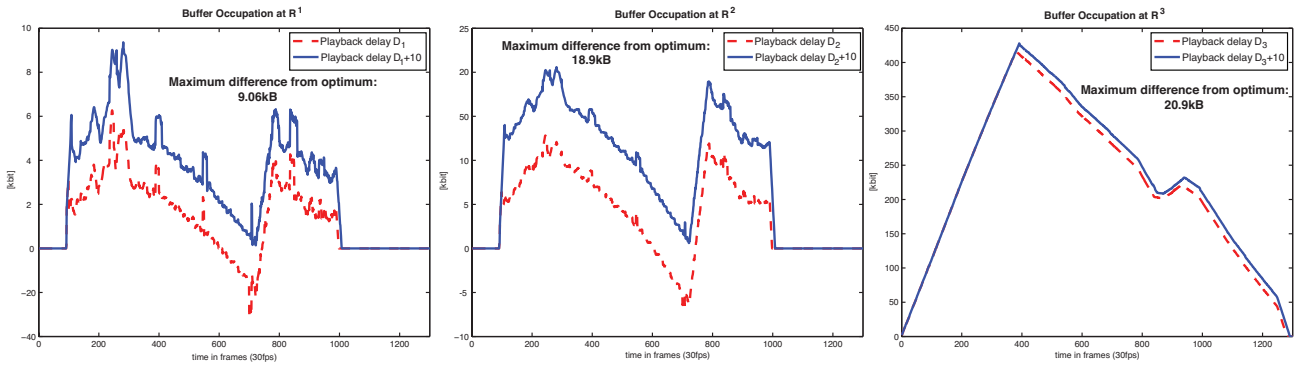

Fig. 15. Receiver buffer occupancy using the rate adaptive streaming algorithm.

[15] P. Thiran and J.-Y. L. Boudec, *Network Calculus*, vol. 2050 of *Lecture Notes on Computer Science*. Springer-Verlag GmbH, 2001.
[16] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 301–317, March 2001.
[17] "Momusys code, MPEG-4 verification model version 18.0." ISO/ IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio, Jan 2001.
[18] T. Stockhammer, H. Jenkac, and G. Kuhn, "Streaming Video over Variable Bit-Rate Wireless Channels," *IEEE Transactions on Multimedia*, vol. 6, pp. 268–277, April 2004.
[19] "Nistnet network emulator." http://www-x.antd.nist.gov/nistnet/.