SCHOOL OF ENGINEERING - STI
SIGNAL PROCESSING INSTITUTE
Christophe De Vleeschouwer, Jacob Chakareski and Pascal Frossard

CH-1015 LAUSANNE

Telephone: +4121 6932601
Telefax: +4121 6937600
e-mail: pascal.frossard@epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

THE VIRTUE OF PATIENCE IN LOW COMPLEXITY SCHEDULING OF PACKETIZED MEDIA WITH FEEDBACK.

Christophe De Vleeschouwer, Jacob Chakareski and Pascal Frossard

Swiss Federal Institute of Technology Lausanne (EPFL)

Signal Processing Institute Technical Report

TR-ITS-2005.022

August 24, 2005

Part of this work has been submitted to IEEE TMM.

This work has been supported by the Swiss NSF, under grant PP002-68737, and by the Belgian NSF.

The virtue of patience in low complexity scheduling of packetized media with feedback.

Christophe De Vleeschouwer, Jacob Chakareski, and Pascal Frossard

Abstract

We consider streaming of pre-encoded and packetized media over best-effort networks in presence of acknowledgment feedbacks. We first review the rate-distortion optimization framework in such scenarios. Given an estimation of future transmission resources, and knowing about past transmissions and received acknowledgments, a scheduling algorithm is defined as a mechanism that selects the data to send over the network at any given time, so as to minimize the end-to-end distortion for the given communication resources. Since the computational complexity induced by optimal solution is unacceptable for practical scenarios, we propose to limit the solution search space in using a greedy scheduling strategy. However, our work highlights the rate-distortion sub-optimality of popular greedy schedulers, which are strongly penalized by anticipated retransmissions. We therefore proposes an original scheduling algorithm that avoids premature retransmissions, while preserving the low computational complexity of the greedy paradigm. Such a scheduling strategy allows to adapt to network QoS fluctuations, with close to optimal rate-distortion performance. Our experimental results demonstrate that the proposed patient greedy (PG) scheduler provides a reduction of up to 50% in transmission rate relative to conventional greedy approaches, and that it brings up to 2dB of quality improvement in scheduling classical MPEG-based packet video streams.

I. INTRODUCTION

The proliferation of high-bandwidth and wireless Internet connections has increased the demand for a low-cost and flexible access to stored media content. Yet, to become a reality, widespread media dissemination has still to face the lack of guarantees offered by the network in terms of bandwidth, delay and error rates. In order to cope with transmission channel fluctuations, the on-demand media server has to implement adaptive streaming solutions, which can be roughly classified into four categories [1]: (i) versioning, with several encodings of the same source at different rates and different error protection levels, but this solution often suffers from coarse granularity, (ii) transcoding within the network, which however often results in computationally complex solutions, (iii) scalable coding of the media stream, and (iv) efficient data packet selection and scheduling, that minimize the end-to-end distortion under stringent timing and channel constraints.

Solutions from the last category are certainly the most generic, and in general offer the possibility of dealing with any media encoding format. They often consider the relative importance of the media data or packets, and adapt to the channel errors and bandwidth fluctuations by selective dropping of data, at the level of frames [2], or layers [3], [4], for example. Most of these works however proceed by greedy rate allocation, and often do not consider neither the dependencies generated by media encoding, nor rate-distortion optimality. In this context, our work addresses the problem of rate-distortion optimized streaming of packetized media over a best-effort packet network, and scenarios with sender-driven (re)transmission using acknowledgement (ACK) feedback are more particularly considered. It precisely targets the definition of appropriate *scheduling* methods to decide which packet should be forwarded to the client at any given time, for an arbitrary packetization of encoded media content. Efficient scheduling considers simultaneously media packet importance, and transmission status, in order to optimize the quality experienced by the streaming client. Such a framework has been nicely formalized in [5], and solutions to the rate-distortion optimized (RaDiO) streaming problem have been proposed in [5], [6], [7]. However, those solutions present a high computational complexity, which make their use impossible in many practical scenarios. They also do not provide any guarantee on the sending rate trace, and thus may not be able to respect instantaneous network bandwidth fluctuations.

The goal of the present work is to propose a streaming solution able to adapt to instantaneous rate constraints, with low computational complexity, and yet close to optimal performance from an end-to-end distortion perspective. We first review an extension of the RaDiO framework, called ARC-RaDiO, that is able to adapt to instantaneous rate constraints, with the introduction of sending buffer. That scheduler, initially proposed in [8], offers optimized rate-distortion performance, but at the price of high computational complexity, even larger than the one of the original RaDiO algorithm. On the other hand, most previous works about low complexity scheduling end up in recommending the implementation of a greedy mechanism to match the instantaneous rate of a connection, while approximating some rate-distortion optimal, generally computationally intractable, solution (e.g., [5], [9], [10], [11]). Related to these works, the study in [12] also considers a rate-distortion approach, but analyses the impact of a very narrow observable horizon on the scheduling algorithm. As such it provides a complementary study to our work. The study is performed in the context of a reliable (no losses) but fluctuating bandwidth channel. To alleviate the adverse impact of the limited search horizon on the end-to-end distortion, authors in [12] propose to penalize the

Christophe De Vleeschouwer is with Laboratoire de Télécommunications, Université catholique de Louvain, Belgium (email: devlees@tele.ucl.ac.be). Jacob Chakareski and Pascal Frossard are with the Signal Processing Institute - LTS4, of Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne (email: jakov.cakareski@epfl.ch, pascal.frossard@epfl.ch). The work was funded by the Swiss NFS, under grant PP-002-68737, and by the Belgian NFS.

expected distortion function by the decoder buffer occupancy. In essence, forcing minimal buffer occupancy helps to decrease the penalty incurred by the variable bit rate channel, when the observable horizon is limited. Our work does not focus on very limited horizon. Conversely to [12], it envisions media streaming over channels that are subject to losses and random delays. Our main contribution consists in deriving computationally tractable, whilst close-to-optimal, scheduling solution for such scenarios.

In this context, our paper demonstrates that the greedy approach is sometimes far from optimal, and defines an original patient greedy (PG) scheduler. We choose to keep up with acceptable computational complexity of greedy approaches, but improve their performance by delaying some packet scheduling decisions. The intuition behind this strategy is that greedy scheduling is often too prompt to re-transmit packets, for which acknowledgements generated by an initial transmission are most likely to arrive at the sender in a very short future. This intuition has been verified experimentally by the number of purely redundant packet retransmissions generated by greedy approaches. The scheduling problem is formalized to take into account the possibility of delaying re-transmissions, and computes the impact of such a decision on the expected rate-distortion performance. The resulting PG scheduler is finally shown to outperform conventional greedy strategies and to reach close to optimal rate-distortion performance, without increasing the computational complexity in the scheduling algorithm.

The rest of the paper is organized as follows. Section II presents the streaming framework considered in this paper, and formalizes the rate-distortion optimization problem for sender-driven media scheduling, with average and instantaneous rate constraints. Section III presents the conventional greedy scheduling mechanisms, generally proposed as a computationally tractable approximation of the RD optimal solution. The sub-optimality of conventional greedy approaches is further examined. In Section IV, we present our novel Patient Greedy packet scheduler and, we examine its performance via simulations in Section V. Finally, we provide concluding remarks in Section VI.

II. Rate-distortion optimal scheduling

A. Channel and media models

In order to formalize the framework of sender-driven adaptive streaming, we follow the abstraction model defined in [5]. It provides a generic and widely accepted formulation that has significantly advanced the state of the art in streaming media systems. The media source is assumed to have been encoded and packetized into a finite set of data units, stored on a media server and abstracted as a group of interdependent data units (GODs). The interdependency between the data units is expressed by a direct acyclic graph, which induces a partial order relation $\prec$ among the data units. We write $l' \prec l$ when data unit l can only be decoded if data unit l' has been decoded. We say that data unit l' (l) is an ancestor (descendant) of data unit l (l'). The l^{th} data unit is characterized by its size S_l in bytes, its importance ΔD_l in units of distortion, and its delivery deadline $t_{D,l}$. The gain ΔD_l in distortion is the amount by which the distortion is decreased if data unit l is decoded, compared to the distortion if only the ancestors of l are decoded.

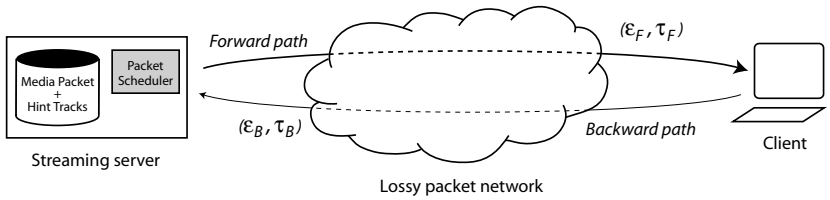


Fig. 1. Sender-driven adaptive streaming with client feedback. Media packet are sent from the streaming server to the client over a lossy packet network, whose channel paths are characterized by random delays τ_F and τ_B, and packet loss probabilities ϵ_F and ϵ_B.

When the streaming server selects a data unit for transmission, the data unit is encapsulated into a packet and sent over the network. A data unit can be encapsulated in more than one packet (i.e., retransmissions are possible), but we assume that a packet contains one and only one data unit. As in [5], the network forwarding path is modeled as an independent time-invariant packet erasure channel with random delays. That means that a packet sent at time t can be either lost with probability ε_F, independent of t, or received at time t', where the delay $\tau_F = t' - t$ is randomly drawn with probability density function p_F. Similarly, when an acknowledgment packet is sent from the client to the server through the backward channel, it is either lost with probability ε_B, or received after a delay τ_B, drawn with probability density function p_B. Each forward or backward packet is lost or delayed independently of other packets. For convenience, to combine the packet loss probability and the packet delay density into a single probability measure, we define a forward (backward) trip time random variable, denoted FTT (BTT), that is assigned to ∞ when the packet is lost, and is set to τ_F (τ_B) when the packet is not lost. The round trip time RTT is a random variable defined as the sum of FTT and BTT. The streaming scenario is illustrated in Figure 1.

B. The RaDiO framework

One of the most popular frameworks proposed in the literature to solve the problem of how and when to transmit a group of interdependent data units in a rate-distortion optimal way is given by the **Ra**te-**Di**stortion **O**ptimization (RaDiO) framework

introduced by Chou and Miao [5], and further studied by Röder et al. [6], [7]. The RaDiO approach also laid down the groundwork for recent studies on streaming media from multiple servers [13], via intermediate proxy servers [14], or in streaming systems with rich client acknowledgments [15], for example. Moreover, the formalism proposed by Chou and Miao is in accordance with other works that consider the scheduling media content over unreliable networks based on rate-distortion optimization techniques [9], [10], [11]. Such a popularity certainly justifies the interest we bring to the implementation of that framework. In this section, we review the RaDiO framework, with a special emphasis on its computational complexity.

To compute rate-distortion optimal transmission schedules for a group of interdependent data units (GOD), the authors in [5] assume independent transmissions of data units, and describe the transmission of L interdependent data units by a policy vector $\vec{\pi} = (\pi_1, ..., \pi_L)$, where π_l, for $l \in \{1, ..., L\}$, denotes the transmission policy associated with the l^{th} data unit. The policy π_l is a binary vector that defines the sender's behavior regarding data unit l at a pre-defined set of time instances $\{t_{l,0}, t_{l,1}, ..., t_{l,N_l-1}\}$, corresponding to the N_l transmission opportunities assigned to data unit l. Hence, $\pi_l = (\pi_l(0), \pi_l(1), ..., \pi_l(N_l-1)) \in \{0,1\}^{N_l}$, where $\pi_l(i) = 1$ means that data unit l should be sent at opportunity i if not yet acknowledged. Associated to the policy π_l, the error $\epsilon(\pi_l)$ defines the probability that data unit l does not reach its destination before its delivery deadline $t_{D,l}$ as

$$\epsilon(\pi_l) = \prod_{i:\pi_l(i)=1} P\{FTT > t_{D,l} - t_{l,i}\} \ . \tag{1}$$

The cost $\rho(\pi_l)$ associated to policy π_l is further defined as the expected number of data unit transmissions, as given by

$$\rho(\pi_l) = \sum_{i:\pi_l(i)=1} \left(\prod_{j<i:\pi_l(j)=1} P\{RTT > t_{l,i} - t_{l,j}\} \right) \ . \tag{2}$$

We note here that the set of policies π_l^* whose cost-error points $(\rho(\pi_l^*), \epsilon(\pi_l^*))$ lie on the lower convex hull of the set of all achievable (ρ, ϵ) points are denoted to be the cost-error optimal policies for data unit l, because they minimize the Lagrangian cost $J_\lambda(\pi_l) = \epsilon(\pi_l) + \lambda\rho(\pi_l)$ for some $\lambda > 0$. They can be computed with a worst case complexity of $O(N_l 2^{N_l})$ [6], which grows exponentially in the number of transmission opportunities.

Regarding the GOD, Chou and Miao [5] define the expected transmission rate $R(\vec{\pi})$ and distortion $D(\vec{\pi})$ associated to $\vec{\pi}$ as

$$R(\vec{\pi}) = \sum_{l=1}^{L} \rho(\pi_l) S_l \ \text{, and} \tag{3}$$

$$D(\vec{\pi}) = D_0 - \sum_{l=1}^{L} \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})) \ , \tag{4}$$

where D_0 denotes the distortion when no data unit has been received in time, S_l is the size of data unit l, and ΔD_l its importance. A convex-hull RD optimal policy vector is then defined as a vector that minimizes the Lagrangian function,

$$J_\lambda(\vec{\pi}) = D(\vec{\pi}) + \lambda R(\vec{\pi}) \ \text{, for } \lambda > 0 \ . \tag{5}$$

A solution to find the RD optimal policy vector $\vec{\pi}$ for a given $\lambda > 0$, and whose complexity grows linearly with the number L of dependent data units, is proposed in [5]. This solution, denoted Iterative Sensitivity Adjustment (ISA), relies on the cost-error optimal policies defined for single data units, and is based on an iterative descent algorithm that minimizes $J_\lambda(\vec{\pi}) = J_\lambda(\pi_1, ..., \pi_L)$ one policy at a time, while keeping the other policies fixed. A pass of L iterations (over the L data units in the GOD) requires at most $O(LS)$ operations, where S denotes the longest dependency path involving one of the L data units of interest[1]. The ISA algorithm has been proven to converge to a local optimum in a small number $n_{i\vec{\pi}}$ of iterations (a few passes, typically two or three, over the L data units). Alternatively, Röder et al. [6] have proposed a branch and bound algorithm to compute a global optimum. However its complexity grows exponentially both in the number of transmission opportunities and the number of data units, which is certainly too much for on-line applications [6].

Under its original formulation, the RaDiO framework presented above leads to an often intractable optimization problem, and a very complex scheduling strategy. In practice, the number of possible streaming strategies has therefore to be constrained in order to bound the search complexity. To limit the number of data units involved in the streaming session at any given time, we follow the authors in [5] and define τ to be the maximal delay accepted for pre-fetching. In other words, at current time t, only those data units whose delivery deadline lies in the sliding window $[t, t+\tau]$ are given the opportunity to be transmitted. The authors in [5] then propose to select the RD optimal policy vector corresponding to an average rate constraint imposed by the channel. Given the average target rate, a bisection search algorithm [16], [17] adjusts λ in Equation (5), to select the policy vector that (i) is convex-hull RD optimal, and (ii) has the largest rate among the optimal policy vectors that respect the rate constraint.

The computational complexity associated to this process can then be estimated as follows. As an initial step, the cost-error optimal policies are computed for the L data units whose delivery deadline lies in the sliding window $[t, t+\tau]$. This is done with

[1]In essence, the longest dependency path represents the largest number of ancestor or descendant data units for a the data unit in the GOD.

a total worst case complexity of $O(L.N2^N)$, N being the number of transmission opportunities for each data unit [6]. For a given Lagrange multiplier λ, the rate-distortion optimal policies are then computed based on the initial set of cost-error optimal policies using the ISA algorithm. This is done in $n_{i\vec{\pi}}$ iterations, with O(L.S) complexity. Then, using a bisection search the value of λ is adjusted and the optimization algorithm is run again to recompute the transmission policies, with the new value of λ. The procedure is repeated till convergence of λ to the value for which the computed optimal transmission policies correspond to the average target transmission rate. Therefore, the overall complexity of the whole procedure is $O(N2^N + n_{i\lambda}n_{i\vec{\pi}}LS)$. It becomes apparent that computing the optimal individual cost-error policies π_l is the major bottleneck of the optimization procedure, with an exponential complexity in terms of the number of transmission opportunities. Therefore, N needs to be kept at a reasonable value so that the system remains computationally feasible. Note that an in-depth analysis of the computational complexity of a RaDiO packet scheduler is proposed in [18].

C. RaDiO with adaptive rate control: the ARC RaDiO

Under its original formulation, the RaDiO optimization problem only satisfies an average rate constraint, which may lead to channel resources requirements that are incompatible with the network policy. In other words, a scheduling mechanism based on the RaDiO framework is in general unable to follow the instantaneous bandwidth fluctuations of a given channel. Since the RaDiO approach measures the distortion-rate performance in an average sense, it is possible for such a system to transmit most of the data units in a single burst, resulting in a large instantaneous rate despite a low average rate. Hence, such behavior might cause a large mismatch between the scheduler and the channel instantaneous rates. A buffer absorbing the mismatch would introduce unnecessary delays in the transmission, resulting in sub-optimality.

That observation is of crucial importance, and it actually urged [5] to propose simplifications of the optimal RaDiO scheduling algorithm that only approximate the RD optimal solution, but which are able to adapt to the dynamic channel instantaneous rate constraints. The λ parameter is adjusted so that exactly one data unit is selected for transmission at each opportunity. Note that the equations provided in [5] reveal that this particular case is in essence equivalent to a conventional greedy scheduling solution, that expectedly stays sub-optimal from a rate-distortion perspective (as explained later in Section III-B). The original RaDiO framework can only fit the channel instantaneous rate constraints at the expense of some loss in RD performance.

In order to cope with that limitation, we now introduce a variation of the original RaDiO framework [5], called ARC RaDiO that is able to meet the instantaneous channel bandwidth constraints while preserving rate-distortion optimality. It specifically takes into account an instantaneous bandwidth constraint imposed by the communication channel over which media packets are sent. The resulting RaDiO with Adaptive Rate Control, is presented in detail in [8].

The ARC-RaDiO accounts for the existence of a sender's buffer located between the RaDiO scheduler and the communication channel, and replaces the conventional constraint on the average transmission rate from the original RaDiO framework, with multiple rate constraints on the buffer level. Specifically, no buffer overflow must be created along the time over which packets are scheduled for transmission by the cumulative rate resulting from the selected policy vectors. The problem is formalized as follows. Beginning at time t_0, the scheduler samples the time with a period equal to T[2]. At time $t_i = t_0 + iT$, the scheduler selects the data units that are pushed to the sender's buffer at time t_i. Following the window-based control paradigm introduced above, at any given time t_i, only those data units l whose delivery deadline $t_{D,l}$ lies in the transmission window $[t_i, t_i + \tau]$ are given the opportunity to be transmitted. Similar to Section II-B, let $\overrightarrow{\pi(t_i)} = (\pi_1, ..., \pi_{L(t_i)})$ denote the vector of transmission policies for the $L(t_i)$ data units contained in the transmission window at time t_i. In practice, for complexity reasons, the transmission opportunities for each data unit is limited to the set of time instants defined by $\{t_i, t_i + T, ..., t_i + (N-1)T)\}$, with $N < \tau/T$. Note that some of these $L(t_i)$ data units might have already been transmitted at previous transmission opportunities. The transmission history of these data units obviously affects their expected error $\rho(\pi_l)$ and expected cost $\varepsilon(\pi_l)$, and is thus taken into account during policy optimization (see [5], [8]).

To formalize the buffer management process, ARC-RaDiO extends the notion of expected cost by incorporating the time at which data are transmitted. Hence, $\rho(\pi_l(j))$ denotes the expected cost under policy π_l at the j^{th} transmission opportunity, i.e., at time $t_i + jT$. At the GOD level, we define $R_j(\overrightarrow{\pi(t_i)})$ to be the expected transmission rate at the j^{th} transmission opportunity, considered at current time t_i. For $j = 0, ..., N-1$,

$$R_j(\overrightarrow{\pi(t_i)}) = \sum_{l=1}^{L(t_i)} B_l \rho(\pi_l(j)) \ . \tag{6}$$

The ARC-RaDiO optimal policy vector at current time t_i is denoted $\overrightarrow{\pi(t_i)}^*$, and is the one that minimizes the expected distortion computed on $L(t_i)$ data units based on Equation (4), and such that the cumulative transmission rates over the N transmission opportunities of interest do not create any overflow of the sender buffer, with available space $BS(t_i)$, and constant draining

[2]The choice of T should be related to the average round-trip time of the underlying communication channel, as it needs to allow the sender to adapt in a timely fashion to the returning acknowledgement packets from the receiver.

rate R_c. Note that the draining rate can be made dynamic and varying over time, without changing the following development. Formally, we have

$$\overrightarrow{\pi(t_i)}^* = \text{argmin}_{\overrightarrow{\pi(t_i)}} D(\overrightarrow{\pi(t_i)}), \tag{7}$$

$$\text{s.t. } \sum_{j=0}^{k} R_j(\overrightarrow{\pi(t_i)}) \leq kTR_c + BS(t_i), \text{for } k = 0, ..., N-1. \tag{8}$$

In particular, we note in Equation (8) that the constraint corresponding to $k = 0$ is a deterministic constraint. This is because the transmission of data units at current time t_i that are recommended by $\overrightarrow{\pi(t_i)}$ does not depend on the (non-)reception of feedbacks in the future. They are effective and define the actual rate at current transmission opportunity. Hence, the constraint imposed by $k = 0$ maintains the bit budget allocated to the current transmission interval below the available capacity. In other words, it forces the scheduler to follow the instantaneous rate offered by the channel at any time.

To simplify notations, starting from here, we omit the dependency on the current time t_i for $\overrightarrow{\pi(t_i)}$ and all related variables. As in [8], the constrained optimization is reformulated as an unconstrained problem using Lagrange multipliers. Hence, we look for the policy vector that minimizes the Lagrangian

$$J_{\vec{\lambda}}(\vec{\pi}) = D(\vec{\pi}) + \sum_{k=0}^{N-1} \lambda_k \sum_{j=0}^{k} R_j(\vec{\pi}) \ , \tag{9}$$

for a vector of positive Lagrange multipliers $\vec{\lambda} = [\lambda_0, ..., \lambda_{N-1}]$. By rearranging Equation (9) and defining $\lambda'_k = \sum_{j=k}^{N-1} \lambda_j$, for $k = 0, ..., N-1$, the Lagrangian becomes

$$J_{\vec{\lambda}'}(\vec{\pi}) = D(\vec{\pi}) + \sum_{k=0}^{N-1} \lambda'_k R_k(\vec{\pi}) \ . \tag{10}$$

Given a vector $\vec{\lambda}'$ of positive and decreasing λ'_k values, the policy vector $\vec{\pi}$ that minimizes the Lagrangian $J_{\vec{\lambda}'}(\vec{\pi})$ is found using an iterative descent algorithm that minimizes $J_{\vec{\lambda}'}(\pi_1, ..., \pi_L)$ one policy at a time, keeping the other policies fixed [8]. The algorithm is similar to the algorithm presented in [5] for the conventional RaDiO system, in that it searches for one single data unit policy at a time. However, its practical implementation appears to be more complex. In the conventional RaDiO case, [5] demonstrates that the RD optimal transmission policy for a single data units, given other data units policies, is directly provided by the cost-error convex-hull that is computed once for all, and with $O(N2^N)$ complexity, for that single data unit at the current time instant. In contrast, because the ARC-RaDiO considers a vector of N λ'_k values, the convex-hull becomes a N-dimensional surface so that the mapping between an arbitrary vector of Lagrangian multipliers and its corresponding optimal policy can not be handle efficiently anymore. For this reason, ARC-RaDiO is forced to perform an exhaustive search over the entire set of policies for a single data each time the iterative algorithm has to optimize that single data unit policy, given the policies for other data units. As a consequence, for a given $\vec{\lambda}'$, the optimal policy vector is found in $n_{i\vec{\pi}}$ iterations over the L data units with a complexity of $O(n_{i\vec{\pi}} LN2^N + n_{i\vec{\pi}} LS)$. In comparison with the RaDiO framework, we note that the dominating factor $L.N2^N$ is now multiplied by the number of iterations $n_{i\vec{\pi}}$.

Now that we have explained how the optimal policy vector can be computed for a given vector $\vec{\lambda}'$ of Lagrange multipliers, we are interested in the selection of a set of Lagrange multipliers that solves the initial problem formulated in (8), and thus results in a (nearly) constant transmission rate equal to R_c over the whole period covered by the N transmission opportunities. The search for a set of appropriate Lagrange multipliers follows an iterative approach inspired by [19], [20] and described in [8]. Let $n_{i\vec{\lambda}'}$ denote the number of iterations to converge to appropriate vector of Lagrange multipliers. Note that, because N constraints have to be satisfied simultaneously, the number $n_{i\vec{\lambda}'}$ of iterations to converge to the right $\vec{\lambda}'$ vector is larger than the number $n_{i\lambda}$ of iterations needed by the conventional RaDiO framework to converge to the optimal λ parameter. Hence, the total complexity of the ARC-RaDiO framework can be bounded with $O(n_{i\vec{\lambda}'} n_{i\pi}(LN2^N + LS))$, and is definitively larger than the conventional RaDiO system complexity.

In general, the excessive complexity requirements of the conventional RaDiO framework and even more of the ARC-RaDiO system, make them quite limited for practical online applications of media packet scheduling. However, they can be still useful in determining bounds on RD performance of practical streaming systems. Therefore, in the rest of the paper, we propose a novel packet scheduling algorithm that exhibits much smaller computation complexity relative to the fully optimized RaDiO systems, as described above, while preserving as much as possible of their end-to-end rate-distortion performance.

III. Towards low complexity scheduling

A. Greedy strategy

The previous sections have clearly outlined that the rate-distortion optimized scheduling mechanisms are too complex to be processed on-line and in real-time, as required by common realistic streaming applications. As discussed earlier, several

authors have addressed the problem of scheduling media content over unreliable networks, and many works have proposed to control streaming systems based on rate-distortion optimization techniques, in formalizing the scheduling decisions as a partially observable Markov decision process [5], [9], [10], [11]. However, in order to limit the computational complexity and/or to match the instantaneous rate imposed by the network, these works generally end-up in recommending the use of heuristic greedy scheduling mechanisms that transmit, at any given time, the data unit that maximizes the decrease in distortion expected per unit of rate. Such a popular mechanism is referred to as conventional greedy mechanism in the rest of the paper, and is generally presented as a degenerated solution of RD optimal scheduling problems. For example, [5] proposes to adjust the λ parameter in Equation (5) so that exactly one data unit is selected at each transmission opportunity offered by the network. This further allows for controlling the instantaneous streaming rate, as discussed before. Other examples of studies that recommend the use of greedy mechanisms are [11] and [21].

More formally, the greedy scheduling mechanism proposed by all these works can be summarized as follows. Let t and τ respectively denote the current time and the maximal pre-fetching delay, so that only those data whose deadlines lie between t and $t+\tau$ are considered for transmission at time t. The set of data units whose delivery deadlines lie between t and $t+\tau$ is denoted Γ_τ^t. At time t, when a packet has to be sent over the network, the greedy approach selects the data unit in Γ_τ^t that maximizes the expected decrease in distortion per unit of rate. Let $\varkappa_l^t$ denote the transmission history for the l^{th} data unit at time t. Specifically, $\varkappa_l^t = \{t_l^1, t_l^2, ..., t_l^{P_l}\}$ defines the P_l time instants at which the l^{th} data unit has been transmitted in the past. We now estimate the probability $p_c^t(l \mid \varkappa_l^t)$ for the l^{th} data unit to be received on-time at the client, knowing about its transmission history $\varkappa_l^t$. When an acknowledgment has been received for data unit l, $p_c^t(l \mid \varkappa_l^t)$ is obviously equal to 1. In absence of an acknowledgment packet for l, we note that data unit l only fails to reach the client on-time when all its transmission attempts fail. Because we assume independent packet transmissions, in absence of an ACK for l, we have therefore

$$p_c^t(l \mid \varkappa_l^t) = 1 - \prod_{i \leq P_l} P\{FTT_i^l > t_{D,l} - t_l^i \mid RTT_i^l > t - t_l^i\} \ , \tag{11}$$

where FTT_i^l and RTT_i^l respectively denote the forward and round trip time random variables associated to the i^{th} (re)transmission of the l^{th} data unit. These random variables have the same distribution as the RTT and FTT variables defined in Section II-A. We now estimate the decrease in distortion β_l^t that can be expected at time t from an additional transmission of the l^{th} data unit. Taking the dependencies among data units into account, we have

$$\beta_l^t = \left[p_c^t(l \mid \varkappa_l^t \cup \{t\}) - p_c^t(l \mid \varkappa_l^t)\right] \times \sum_{l' \succeq l} \left(\Delta D_{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'' \mid \varkappa_{l''}^t) \right) \ . \tag{12}$$

The sum in Equation (12) reflects the fact that the reception of the l^{th} data unit is beneficial for l but also for all of its descendants. The product in Equation (12) expresses the fact that correct decoding of data unit l' is subject to on-time reception of all its ancestors. At current time t, the greedy approach (re)transmits the data unit, denoted $l^G(t)$, that maximizes the expected gain in distortion per unit of rate. Therefore, we have

$$l^G(t) = \underset{l \in \Gamma_\tau^t}{\operatorname{argmax}} \frac{\beta_l^t}{S_l} \ . \tag{13}$$

In terms of complexity, the greedy algorithm is dominated by the computations involved in Equation (12). They typically correspond to a total of $O(LS+L)$ operations, with S denoting the length of the longest dependency path containing one of the L data units contained in Γ_τ^t, as described earlier. In particular, the term LS corresponds to the computation of the importance for each of the L data, while L accounts for the search for the data unit with the highest rate-distortion importance. Here, we neglect the complexity associated to Equation (11), because it only requires a number of operations per data unit that is equal to the number of past transmissions for that data unit, typically 0 or 1. Since nothing comes for free, greedy strategies have however to pay a penalty in terms of rate-distortion performance.

B. Complexity versus optimality trade-off

It is quite obvious that limiting the strategy search space can only bring a penalty on the optimality of the solution. Such a distortion penalty is however difficult to appreciate. In order to further understand the sub-optimality paid by greedy solutions, we propose to analyze a short representative example. This section presents a comparative analysis of two rate controlled scheduling solutions, the ARC-RaDiO (RD) and the conventional greedy (G) mechanisms, respectively defined in Sections II-C and III-A. The schedulers behaviors are compared in a toy example, but the purpose is to derive appropriate and generic heuristics to improve the conventional greedy scheduling mechanism.

The illustrative scenario uses a sequence of identical media frames composed of 5 hierarchical layers, defined such that $S_{l+1} = S_l = 50$ bits and $\Delta D_{l+1} = \Delta D_l/2$, $\forall l \leq 5$. The frames are encoded at 20 fr/sec. The maximal pre-fetching delay τ is set to 1 sec. The channel delay pdf is modeled as shifted exponential with mean $\mu_F = \mu_B = 180ms$. The channel loss rate is defined by $\varepsilon_F = 0.2$, and $\varepsilon_B = 0$, and the channel rate is set to 6500 bits/sec (=325 bits/frame). Table I compares the

Layer	% on-time		Average # of trans.		Average # of trans. while ACK to come	
	RD	G	RD	G	RD	G
1	100	100	1.41	2.56	0.08	1.27
2	100	100	1.39	2.19	0.07	0.91
3	100	100	1.36	1.26	0.06	0.15
4	100	38	1.26	0.49	0.02	0.02
5	97	0	1.07	0	0.00	-

TABLE I

STATISTICAL COMPARISON OF ARC-RADIO (RD) AND GREEDY (G) SCHEDULING MECHANISMS. MEDIA CONTENT AND CHANNEL CONDITIONS ARE DEFINED IN THE TEXT.

statistics of both greedy and RD optimal algorithms and presents (i) the percentage of data units that have reached the client on-time, (ii) the average number of transmissions per data unit, and (iii) the average number of unnecessary retransmissions, for which an acknowledgment triggered by a previous transmission was on the way to reach the sender before the delivery deadline of the corresponding data unit. For both algorithms, these statistics are presented as a function of the layer index. We observe that greedy solutions might result in significantly suboptimal rate-distortion trade-offs. In particular, the greedy algorithm always fails to transmit the fifth layer because it retransmits too much data from the other layers. Based on the last column in Table I, we conclude that a lot of these retransmissions would be avoided if the sender was more patient in triggering retransmissions: the RD optimal scheduling almost never re-transmit a packet when an acknowledgement is on its way back to the sender, while the greedy scheduling wastes about one retransmission for each packet from the first two layers.

This observation is fundamental, and lays the groundwork for the definition of a novel patient greedy algorithm in Section IV. In short, Table I suggests that the greedy scheduler should wait longer between successive retransmissions, so that the ACKs triggered by previous transmissions of the same data unit do have an opportunity to reach the sender. Therefore, in section IV, we evaluate the advantage stemming from a postponed retransmission, and propose to constrain the conventional greedy algorithm to prevent the retransmission of data units for which a delayed retransmission is likely to bring a benefit in a rate-distortion sense. The resulting patient greedy scheduling solution preserves the simplicity offered by the conventional greedy scheduler since the search space stays limited. However, it significantly improves its rate-distortion performance by considering a larger horizon, with the most probable future packet receptions.

IV. PATIENT GREEDY SCHEDULING

Based on the analysis presented in Section III-B, we now propose a scheduling algorithm that prevents premature retransmissions. In short, Section IV-A analysis the impact of postponing a retransmission. Based on this analysis, Section IV-B and IV-C define two versions of our proposed patient greedy algorithm, which only differentiate in their on-line estimation of the data units probabilities of arrival. Finally, Section IV-D considers practical implementation issues and evaluates the computational complexity of the proposed algorithm.

A. Consequences of a delayed transmission

Before going into the details of the proposed patient greedy scheduling algorithm, we now formally consider the possibility to delay the retransmission of a data unit, and its respective impact on the rate-distortion formulation. Let $\beta_l^{t,t'}$ denote the expected decrease in distortion estimated at current time t for the (re)transmission of the l^{th} data unit at time $t' \geq t$. Similar to Equation (12), we have

$$\beta_l^{t,t'} = \left[p_c^t(l \mid \varkappa_l^t \cup \{t'\}) - p_c^t(l \mid \varkappa_l^t)\right] \times \sum_{l' \succeq l} \left(\Delta D_{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'' \mid \varkappa_{l''}^t) \right) \ . \tag{14}$$

On the right hand side of Equation (14), only $p_c^t(l \mid \varkappa_l^t \cup \{t'\})$ depends on t'. Based on Equation (11), and given the transmission history $\{t_l^i\}_{i \leq P_l}$, in absence of an ACK packet for data unit l by time t, we have

$$p_c^t(l \mid \varkappa_l^t \cup \{t'\}) = 1 - P\{FTT_{P_l+1}^l > t_{D,l} - t'\} \prod_{i \leq P_l} P\{FTT_i^l > t_{D,l} - t_l^i \mid RTT_i^l > t - t_l^i\} \ , \tag{15}$$

which shows that $p_c^t(l \mid \varkappa_l^t \cup \{t'\})$, and consequently the benefit in distortion $\beta_l^{t,t'}$ decreases as t' increases. Furthermore, because an ACK might be received between t and t', the cost in rate associated to a postponed transmission also decreases as t' increases. Formally, we introduce the expected cost $\zeta_l^{t,t'}$ estimated at t and associated to the transmission of the l^{th} data unit at time $t' \geq t$. Given the transmission history $\{t_l^i\}_{i \leq P_l}$ of data unit l, we have

$$\zeta_l^{t,t'} = S_l \prod_{i \leq P_l} P\{RTT_i^l > t' - t_l^i \mid RTT_i^l > t - t_l^i\} \ . \tag{16}$$

B. PG: the patient greedy algorithm

Based on Section IV-A, we know that postponing the retransmission of the l^{th} data unit has a positive impact on the rate consumption, i.e., $\zeta_l^{t,t'}$ decreases as t' increases. But it has a negative impact on the media quality, i.e., $\beta_l^{t,t'}$ decreases when t' increases. To estimate whether the gain in rate is worth the loss in quality, we introduce the Lagrangian factor $\lambda(t)$, which balances the expected gain in rate versus distortion. In particular, $\lambda(t)$ defines the decrease in distortion that can be expected per additional unit of rate at time t. We explain in Section IV-D how $\lambda(t)$ is estimated in practice. Given the Lagrangian factor $\lambda(t)$, we can figure out whether postponing the retransmission from t to t' is likely to bring a global benefit in a rate-distortion sense. For the l^{th} data unit, delaying transmission is beneficial when $\lambda(t)[\zeta_l^{t,t} - \zeta_l^{t,t'}] > \beta_l^{t,t} - \beta_l^{t,t'}$ or, equivalently, when

$$-\beta_l^{t,t} + \lambda(t)\zeta_l^{t,t} > -\beta_l^{t,t'} + \lambda(t)\zeta_l^{t,t'} \ . \tag{17}$$

Based on Equation (17), we say that a data unit is *eligible* for transmission at current time t if there is no global RD benefit to expect from a postponed transmission. Formally, the l^{th} data unit is eligible at time t if

$$t = \underset{t' \in [t, t_{D,l}]}{\text{argmin}} \left(-\beta_l^{t,t'} + \lambda(t)\zeta_l^{t,t'} \right) \ . \tag{18}$$

We now define our proposed Patient Greedy (PG) scheduling mechanism as a greedy scheduling that is constrained to select the data to transmit among the set of eligible data units. Formally, let Ψ_τ^t denote the set of eligible data units contained in Γ_τ^t, and let $l^{PG}(t)$ denote the index of the data unit selected at time t by the PG algorithm. By definition $\beta_l^{t,t} = \beta_l^t$ and $\zeta_l^{t,t} = S_l$. So, similarly to Equation (13), we have

$$l^{PG}(t) = \underset{l \in \Psi_\tau^t}{\text{argmax}} \frac{\beta_l^t}{S_l} \ . \tag{19}$$

The practical selection of eligible data units in Γ_τ^t is considered in Section IV-D. Before we explain how PG is refined when some a priori knowledge is available regarding the data units probabilities of arrival.

C. PG-AL: An extension of PG exploiting a priori probabilities of arrival

Section IV-B has shown that the patient greedy algorithm is based on the estimation at current time t of the decrease in distortion expected for the (re)transmission of the l^{th} data unit at time $t' \geq t$. Based on Equation (14), it also appears that the benefit to expect from the l^{th} data unit (re)transmission largely depends on the probabilities of arrival for the other dependent data units, either ancestors or descendants of l. In particular, to define PG, Equation (14) estimates the probability that a data unit $k \neq l$ reaches the client on-time by $p_c^t(k \mid \varkappa_k^t)$, without any prior information, but only based on the transmission history of the data unit k. In this section, we propose an alternative estimation of the probabilities of arrivals for data units $k \neq l$, which in turns results in the definition of the Patient Greedy algorithm based on Arrivals Likelihood (PG-AL).

At current time t, $p_c^t(k \mid \varkappa_k^t)$ might provide a poor estimation of the actual probability of arrival for k, mainly because possible future transmissions of k are ignored by the transmission history $\varkappa_k^t$. As a complement to $p_c^t(k \mid \varkappa_k^t)$, we consider $p_a^t(k)$ to be the estimation at time t of the *a priori* probability that the k^{th} data unit reaches the client on-time. In practice, $p_a^t(k)$ is estimated based on the fact that streamed content generally corresponds to a sequence of independent GODs characterized by the same acyclic dependency graphs. Under that assumption, $p_a^t(k)$ is defined as the exponentially weighted average of the probabilities of arrival computed *a posteriori* for data units that have already passed their deadline at time t, and that correspond to data unit k in their respective GOD.

Based on the $p_c^t(k \mid \varkappa_k^t)$ and $p_a^t(k)$ estimates, the arrival likelihood (AL) for data unit k is denoted $p_{AL}^t(k \mid \varkappa_k^t)$ and defined by

$$p_{AL}^t(k \mid \varkappa_k^t) = \max(p_c^t(k \mid \varkappa_k^t), \gamma \ p_a^t(k)). \tag{20}$$

In Equation (20), $\gamma \ p_a^t(k)$ is supposed to provide a lower bound for the probability that k reaches the client on-time. In practice, γ results from a trade-off. On the one hand, it has to be large enough, so that $\gamma \ p_a^t(k)$ can alleviate the adverse impact of under estimating the probability of arrival based on $p_c^t(k \mid \varkappa_k^t)$. On the other hand, it should not be too large, so that the lower bound provided by $\gamma \ p_a^t(k)$ stays reliable. Our experiments have shown that a value of $\gamma = 0.5$ reasonably meets these two requirements. So, γ has been set to 0.5 in the rest of the paper. The benefit expected from the (re)transmission of the l^{th} data unit at time $t' \geq t$ that is estimated based on the arrival likelihoods is denoted $\beta_{l,AL}^{t,t'}$, and is defined as follows

$$\beta_{l,AL}^{t,t'} = \left[p_c^t(l \mid \varkappa_l^t \cup \{t'\}) - p_c^t(l \mid \varkappa_l^t)\right] \times \sum_{l' \succeq l} \left(\Delta D_{l'} \prod_{l'' \preceq l', l'' \neq l} p_{AL}^t(l'' \mid \varkappa_{l''}^t) \right) \ . \tag{21}$$

The patient greedy algorithm resulting from Equation (21) is referred to as PG-AL in the rest of the paper. The simulations presented in Section V demonstrate that PG-AL significantly outperforms PG in streaming scenarios for which only a small number of GODs are considered for transmission at a given time t. In other typical scenarios, PG and PG-AL achieve similar performances.

D. Practical implementation considerations

This section explains (i) how to estimate the $\lambda(t)$ parameter defined in Equations (17) and (18), and (ii) how the eligibility condition defined in Equation (18) is verified in practice. It also reveals that PG and PG-AL have a global complexity that is equivalent to greedy scheduling mechanisms.

Regarding the Lagrangian factor $\lambda(t)$, we observe that the rate spared in postponing the retransmission of a data unit is used to (re)transmit one or several additional data unit(s). Obviously, it is only beneficial to spare some rate by postponing the (re-)transmission of a data unit if the resulting loss in distortion is smaller than the gain expected from additional transmissions of other well-chosen data units. Fundamentally, the Lagrange multiplier has to reflect this piece of evidence, by converting the bit savings in terms of the gain in distortion due to additional data transmissions.

Following the above discussion, we have decided to estimate the factor $\lambda(t)$ based on the history of the streaming session. Specifically, $\lambda(t)$ is estimated as the smallest expected benefit per unit of rate observed among the data units that have been sent over the network in a recent past. Doing so, $\lambda(t)$ provides a good estimate of the benefit brought by an additional unit of rate. This is because, following the greedy approach principle, the additional (re)transmission are selected as the ones that are expected to bring the largest benefit per unit of rate among transmissions that have not been considered yet.

In practice, $\lambda(t)$ is defined to be a piecewise constant function, updated at regular time intervals. Let $\{\upsilon_k\}_{k\geq 0}$ denote the sequence of time instants at which $\lambda(t)$ is updated, and let υ_k^- and υ_k^+ denote the instants immediately preceding and following υ_k. We also define λ_k to be the smallest expected benefit per unit of rate encountered among the data units sent during the $[\upsilon_{k-1}, \upsilon_k]$ time interval. The piecewise function $\lambda(t)$ is then derived based on the sequence $\{\lambda_k\}_{k>0}$. Starting with an initial value of $\lambda(\upsilon_0)$ equal to zero, we update the Lagrangian factor based on an exponentially weighted average. We have

$$\lambda(\upsilon_k^+) = \alpha\lambda_k + (1-\alpha)\lambda(\upsilon_k^-) \quad \forall k > 0 \ . \tag{22}$$

To complete Equation (22), we still have to define the parameter α and the sequence of time instants $\{\upsilon_k\}_{k\geq 0}$ at which $\lambda(t)$ is updated. For that purpose, we introduce the notion of self-contained groups of interdependent data units (SGOD). A SGOD is defined so that it does not have any ancestor or descendant among data units that are outside the group. Typical examples of SGOD are a group of pictures in the MPEG terminology, or a frame in the J2K terminology. We propose to update $\lambda(t)$ each time a self-contained group becomes obsolete, i.e., when all data units contained in the group have passed their delivery deadlines. We have chosen to synchronize the $\{\upsilon_k\}_{k\geq 0}$ sequence with the delivery deadlines of SGODs because they occur at regular time intervals, and because we expect some consistency between the smallest expected benefit per unit of rate observed in such intervals. In our simulations, the parameter α has been chosen equal to 0.4, but the function $\lambda(t)$ appears to be quite insensitive to the α parameter because successive values of λ_k are indeed close to each other.

Given $\lambda(t)$, we now explain how the eligibility condition defined by Equation (18) is checked in practice. Because patient greedy algorithms search for the eligible data unit that has the largest β_l^t/S_l ratio (see Equation (19)), the eligibility is tested for data units ordered in decreasing order of β_l^t/S_l ratio, until the first eligible data unit is met. Hence, only L' eligibility tests have to be performed, with $L' << L$. For each test, only a finite number F of t' values are considered. In our simulations, the possible values of t' are distributed regularly between the current time t and the data unit delivery deadline $t_{D,l}$. We have chosen to use the average time elapsed between successive packet transmissions in the recent past as the interval between two successive investigated t' values. It is worth noting that the scheduling system is not sensitive to the sampling period of t'. Indeed, the postponed transmission alternatives are investigated to check whether waiting before retransmission is worthwhile or not. The purpose is thus not to find the exact time t' that minimizes $-\beta_l^{t,t'} + \lambda(t)\zeta_l^{t,t'}$. As a conclusion, it is certainly possible to design an efficient patient greedy scheme that investigates a smaller number of t' values than the one studied in our simulations. However, for each t' values, $\beta_l^{t,t'}$ and $\zeta_l^{t,t'}$ can be computed with a single multiplication and division operation, without referring to ancestors and descendants of l (see Equations (14) and (16)). So, the complexity associated to the eligibility test is O(L'F), which appears to be neglectable in front of the O(L S+L) complexity needed to compute the β_l^t and ζ_l^t values in greedy algorithms. Specifically, we have observed that our implementations of PG and PG-AL had about the same running time (on a PC) than the corresponding implementation of the conventional greedy algorithm. It makes any additional complexity optimization regarding the eligibility test unnecessary. It also confirms that the complexity of PG and PG-AL is dominated by the $O(LS + L)$ term, which is orders of magnitude smaller than the ARC-RaDiO complexity, for typical streaming application scenarios.

V. Simulation results

A. Preliminaries

This section compares the RD optimal rate adaptive scheduling algorithm, i.e. the ARC-RaDiO algorithm, with computationally simpler conventional and patient greedy mechanisms. It considers both artificial data, and classical MPEG-like video packets. The artificial data units correspond to a sequence of identical and temporally equidistant frames, which are decoded independently of each others. Streaming a strictly formatted content provides two major advantages. First, it makes the results easy to reproduce and compare with other contributions. Second, it facilitates the interpretation and understanding of the

underlying scheduling mechanisms as they are not affected by the fluctuations of the media features along time. While being helpful to understand the scheduling behavior, formatted media content is not fully sufficient to apprehend all the components of an actual streaming system. For this reason, streaming sessions that are based on real video content are also reported. They confirm the lessons drawn based on formatted content, but additionally reveal the importance of taking the arrival likelihood into account as proposed in the PG-AL algorithm.

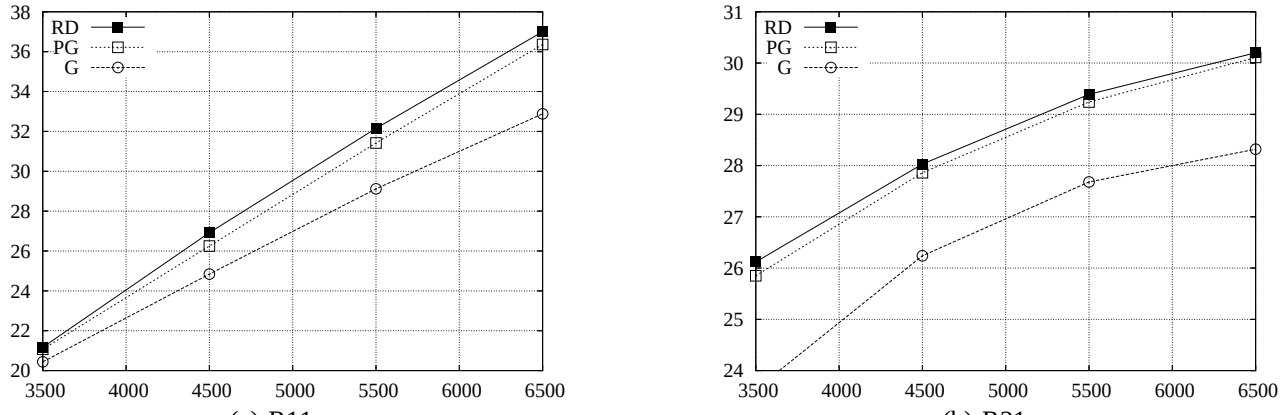
In our simulations, the streaming conditions are similar for both data types. Forward (F) and backward (B) paths are modeled as independent time-invariant packet erasure channels with random delays (see Section II-A) and constant bandwidth. The probability density functions p_F and p_B are modeled as a shifted exponential with mean μ_F (μ_B) and shift $\kappa_F = \mu_F/2$ ($\kappa_B = \mu_B/2$)[5], [21]. In all our simulations, we consider the streaming of a sequence of temporally equidistant frames, and the maximal pre-fetching delay τ is defined in terms of the number of frames W_τ contained in a time interval of τ seconds.

In summary, the results in the reminder of this section demonstrate that the proposed patient greedy algorithms outperform conventional greedy solutions, while reaching close to RD optimal performance. They also show that the distance to which the scheduler can look into the future, defined by the pre-fetching window W_τ, has a significant impact on the streaming performance. For both RD optimal and patient greedy algorithms, the quality improves as the pre-fetching delay increases. In contrast, the conventional greedy algorithm is not able to exploit the flexibility offered by a larger pre-fetching window. Our simulations also reveal that taking the arrival likelihood into account in the patient greedy algorithms is beneficial only when a small number of independent GODs can be pre-fetched simultaneously. In other cases, PG and PG-AL achieve identical rate-distortion performance. Finally, we observe that the patient greedy performs closer to the RD optimal algorithms as the pre-fetching delay increases.

B. Formatted and artificial content

Here, we compare the performances of greedy (G), patient greedy (PG) and ARC-RaDiO (RD) algorithms, based on streaming of formatted media content[3]. In these simulations, the frame rate is 20 fps, and all (patient) greedy approaches use a maximal pre-fetching delay τ equal to 1 sec, i.e., $W_\tau = 20$. Each frame is composed of $N = 5$ data units organized in a hierarchy of layers. All data units have the same size, set to 50 bits. The increase in quality (or equivalently the decrease in distortion) associated to a data unit is defined in units of quality. It only depends on the data unit layer index, and obeys a predefined distortion template, characterized by a constant ratio between the decrease in distortion provided by consecutive layers. Let ΔD_l denote the decrease in distortion for the l^{th} layer. We denote $R11$ the template for which $\Delta D_1 = 8$ and $\Delta D_{l+1} = \Delta D_l$. We denote $R21$ ($R12$) the template for which $\Delta D_1 = 16$ ($\Delta D_1 = 1$) and $\Delta D_{l+1} = \Delta D_l/2$ ($\Delta D_{l+1} = 2\Delta D_l$). For all templates the quality achieved in absence of any data unit is set to 0. Note that the $R11$ and $R21$ templates are certainly the most realistic, as real world media coders generally encode the most important information in the first layers.

Figure 2 considers a channel with reliable feedback (no loss on the reverse path), and presents the average quality as a function of the forward channel bit-rate for the greedy (G), the patient greedy (PG), and the rate-distortion optimal ARC-RaDiO (RD) algorithms. For each frame, the quality is computed as the sum of the increases in quality provided by decodable data units. The increase in quality associated to a data unit is defined by the distortion template. Figures 2 (a) and (b) respectively consider the $R11$ and $R21$ distortion templates. We observe that PG significantly outperforms G, and achieves performances that are close to the RD ones. Extended simulations with a large range of channel parameters have confirmed that observation.



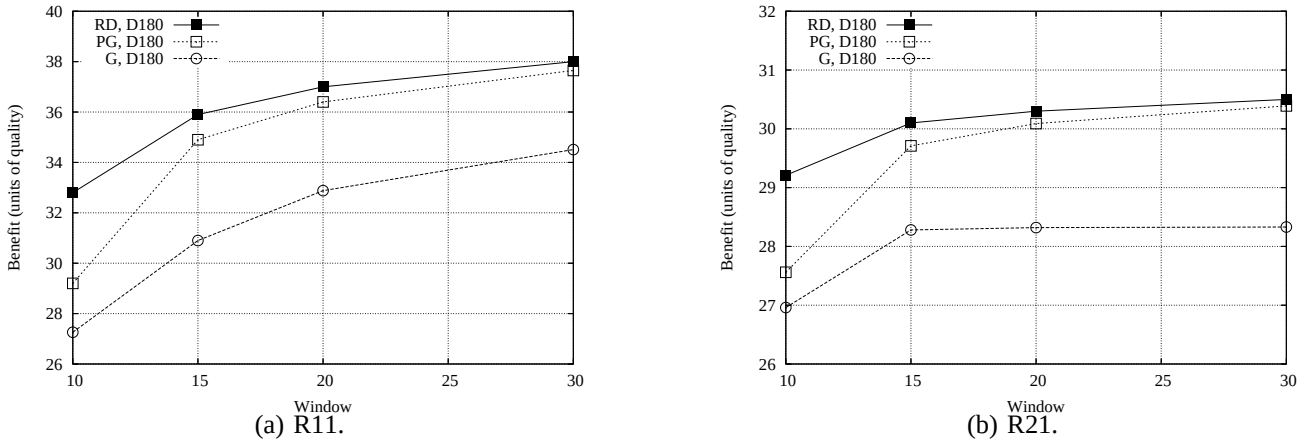
(a) R11. (b) R21.

Fig. 2. Streaming performance (in units of quality) versus channel bit-rate (in bits/sec), for different distortion templates (with $\mu_F = \mu_B = 180ms$, $\varepsilon_F = 0.2$, and $\varepsilon_B = 0$, and $W_\tau = 20$.)

[3]We do not present the PG-AL curves, because they coincide with PG for all the simulations considered in this section.

Figure 3 analyzes the impact of the pre-fetching window size W_τ on the received quality, at a constant channel bandwidth of 6500 bits/sec, and for the R11 and R21 distortion templates. The transmission delays are still defined by $\mu_F = \mu_B = 180ms$ and the maximal pre-fetch delay is defined in terms of W_τ by $\tau = W_\tau \times 50ms$. We observe that, for small values of W_τ, the streamed quality significantly increases with W_τ. When W_τ becomes sufficiently large, the end-to-end quality tends to saturate and does not much improve anymore. In both graphs, we note that for small values of W_τ, PG performs worse than the RD optimal scheduling. In other words, PG needs a sufficient pre-fetch delay to achieve close to RD optimal performance. This sufficient pre-fetch delay is related to the time needed to get a feedback from the channel.

Typically, in Figure 3, PG needs a window W_τ larger than 15 to 20 frames, which corresponds to a maximal pre-fetch delay of about 750 to 1000 ms. The average RTT considered in these simulations is $2 \times 180ms = 360ms$. We conclude that PG needs a pre-fetch delay of about 2 or 3 RTTs before being able to compete with ARC-RaDiO. In most realistic cases, this is not an issue, especially because ARC-RaDiO also needs 2 to 3 RTTs of pre-fetch delay to saturate in quality. Finally, we also observe that the performance improvement to get from an increased pre-fetch delay is larger for patient greedy or buffered RaDiO approaches than for the conventional greedy algorithm. This is not a surprise because both the buffered RaDiO and the patient greedy approach try to optimize the distribution of retransmissions over the entire period of time available before a data unit deadline expires. Conversely, the conventional greedy mechanism retransmits the most critical data without necessarily exploiting the time available to get a feedback about previous transmissions of that data unit.



(a) R11. (b) R21.

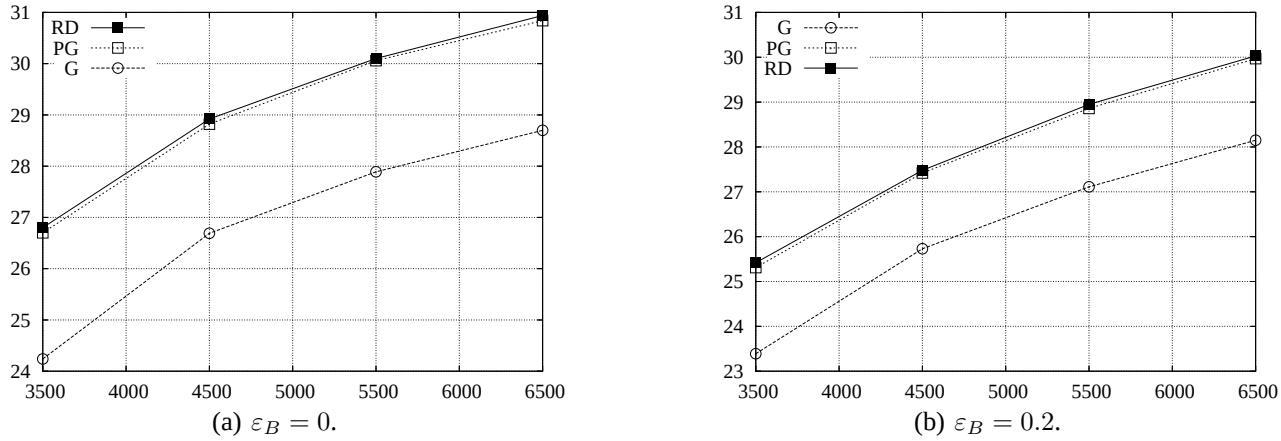
Fig. 3. Performance of the streaming algorithms (in units of quality) versus maximal pre-fetch window size W_τ, with $\mu_F = \mu_B = 180ms$, $\varepsilon_F = 0.2$, and $\varepsilon_B = 0$ and $Rate = 6500bits/sec$.

In Figure 4, we consider a shorter transmission delay ($\mu_F = \mu_B = 60ms$), and the $R21$ distortion template. In both Figures, $\varepsilon_F = 0.2$, but $\varepsilon_B = 0$ in Figure 4 (a) while it is equal to 0.2 in Figure 4 (b). Again, we observe that PG outperforms G, and reaches close to optimal performances. By comparing Figure 4 (a) and Figure 4 (b), we conclude that the sub-optimality of the greedy algorithm is even more significant in presence of reliable acknowledgment feedbacks. Intuitively, this can be understood by the fact that the main drawback of the greedy approach is that it does not wait for an ACK packet that is on the way to the sender, before triggering a retransmission. In contrast, both the PG and RD approaches postpone retransmissions when a benefit can be expected.

To evaluate the gain of PG in terms of rate consumption, Figure 5 plots the ratio between the rates consumed by G and PG to achieve the same average quality as a function of ε_F. The targeted quality is the one obtained by PG at 4.5 kbits/sec. Figure 5 (a) considers a lossless backward channel, and compares the G and PG for different layer distortion templates. We observe that the amount of transmission rate saved by the PG approach is highly dependent on the way quality is allocated among layers. Some distortion templates like $R11$ end-up in relatively good behavior of the greedy algorithm, while others like $R12$ or $R21$ cause a lot of penalizing anticipated retransmissions. Figure 5 (b) analyzes the impact of feedback reliability on the gain provided by the PG algorithm. Losses are either symmetric ($\varepsilon_B = \varepsilon_F$) or one-way ($\varepsilon_B = 0$). We conclude that PG is even more beneficial with reliable feedback, which makes sense as the main PG achievement is a better usage of received ACKs.

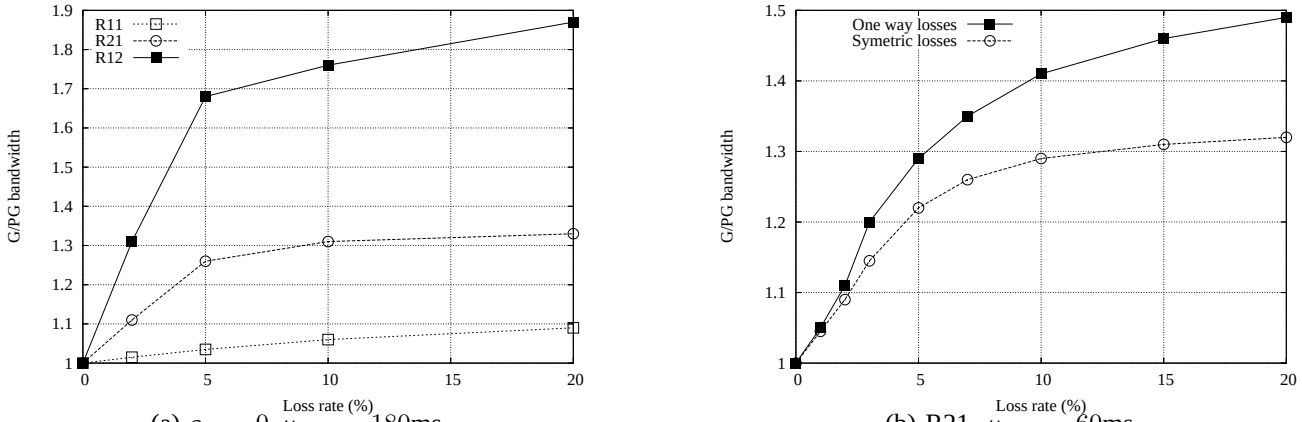
C. Video data

This section compares the rate-distortion performance obtained when streaming packetized video content based on greedy and ARC-RaDiO scheduling algorithms. All video simulations consider the QCIF *Foreman* sequence encoded at 10 fps. The first 120 frames of the *Foreman* sequence are coded using JM2.1 of the JVT/H.264 compression standard, with a constant quantization parameter, for an average Y-PSNR of 35.9 dB and an average rate of 64 kbps. The 120- frame video segment is divided into 6 group of pictures (GOPs), each GOP being composed of an I frame followed by 19 consecutive P frames.



(a) $\varepsilon_B = 0$. (b) $\varepsilon_B = 0.2$.

Fig. 4. Performance of the streaming algorithm (in units of quality), versus channel bitrate (in bits/sec), with $\mu_F = \mu_B = 60ms$, $\varepsilon_F = 0.2$, $R21$ distortion template, and $W_\tau = 20$.



(a) $\varepsilon_B = 0, \mu_{F,B} = 180$ms. (b) R21, $\mu_{F,B} = 60$ms.

Fig. 5. Ratio between the G and PG transmission rates needed to achieve a constant quality, as a function of the loss probability ε_F, with $W_\tau = 20$.

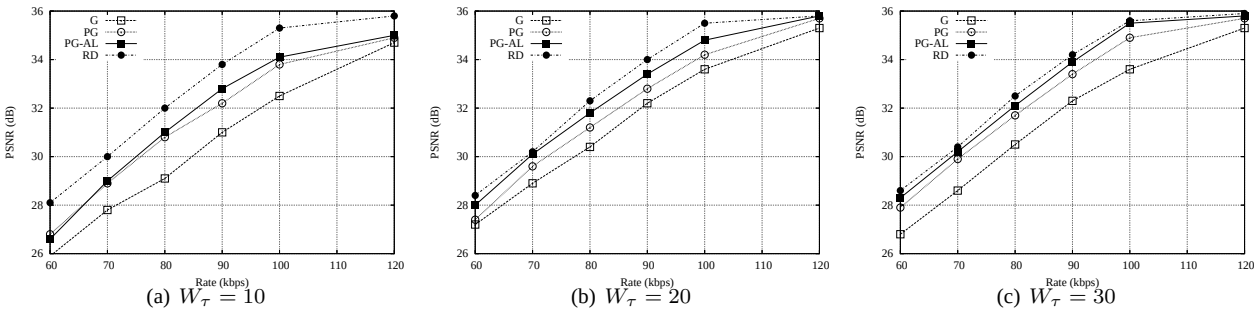
All our simulations consider the concatenation of 20 video segments, each segment being composed of the 120 frames of the Foreman sequence. The purpose of running simulations on a longer concatenated sequence is to alleviate the impact of the probabilistic channel behavior on the average reconstructed PSNR values. We have also observed that both patient greedy and ARC-RaDiO algorithms are more stable than G w.r.t. probabilistic fluctuations of the communication channels, i.e., the PSNR values measured on distinct segments fluctuate less for PG and ARC-RaDiO than for G. In all cases, the play-out delay is equal to 1 second. The pre-fetch delay τ is defined in terms of the number of frames W_τ the scheduler can access in advance. For example, as 10 frames are encoded per second, a pre-fetching window W_τ equal to 20 corresponds to a pre-fetch delay of 2 seconds.

Error concealment of missing video frames has been considered at the client. The concealment works as implemented in the JM2.1 decoder of JVT/H.264. On the scheduler side, the concealment is approximated as proposed in [22], i.e., by assuming that a non-decodable frame is reconstructed based on the last decodable frame. Here, a frame is said to be decodable when the frame and all its ancestors have been received on-time. No concealment is considered between consecutive segments.

In practice, the concealment strategy only affects the benefit to expect from the reception of a data unit, but does not change the behavior of the scheduling algorithms once this benefit has been estimated. We refer the reader to Appendix A for a detailed description of how the scheduler formalizes error concealment and computes the corresponding expected benefit values. In the following, we compare the different scheduling mechanisms, given that the concealment is taken into account to compute the expected benefits, as described in Appendix A.

Figure 6 first considers a symmetric transmission channel, and presents the average luminance (Y) PSNR value of the decoded video frames at the receiver as a function of the forward channel bandwidth for the greedy (G), the patient greedy (PG), the patient greedy with arrival likelihood (PG-AL), and the ARC-RaDiO (RD) algorithms. Figure 6 (a), (b) and (c) respectively consider a pre-fetching window W_τ of 10, 20, and 30 frames.

In all graphs, we observe that the conventional greedy algorithm performs significantly worse than RD optimal or patient greedy scheduling mechanisms. We also observe that PG-AL performs better than PG. So, for video data, it is beneficial to



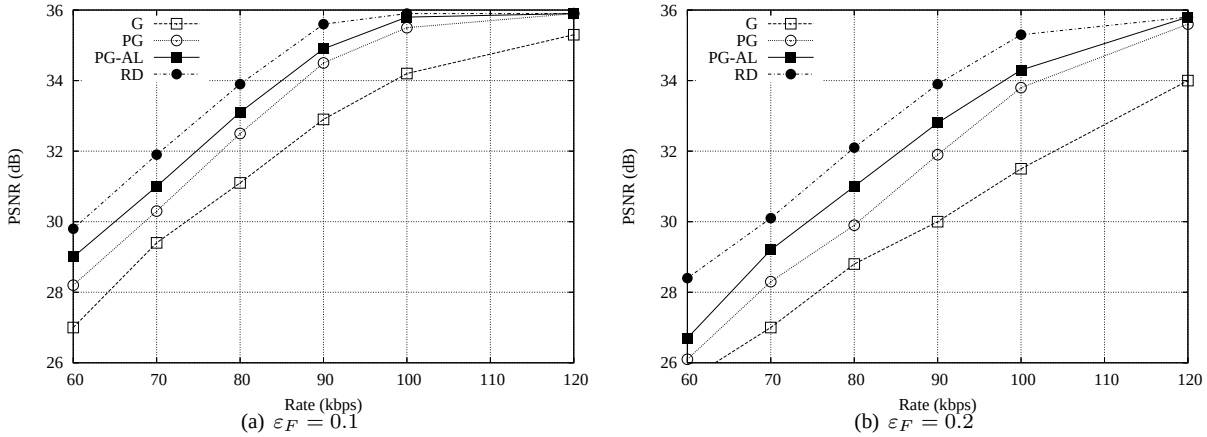
(a) $W_\tau = 10$ (b) $W_\tau = 20$ (c) $W_\tau = 30$

Fig. 6. PSNR quality versus bitrate, when streaming the QCIF Foreman video sequence. The channel conditions are defined by $\mu_F = \mu_B = 100ms$, $\varepsilon_F = \varepsilon_B = 0.1$. Channel bandwidth is constant along the time.

exploit the *a priori* information inferred about the probabilities of arrival, based on previous GOPs transmissions. We explain this benefit as follows. The 20 frames GOP covers a time interval equal to 2 seconds, which is of the same order of magnitude as the authorized pre-fetch delay. As a consequence, only one or two GOPs are considered for transmission simultaneously. Besides, due to the dependency defined within a GOP, once a frame is expected to be lost, its subsequent P frames are expected to be impossible to decode. So, greedy mechanisms, such as G and PG, which estimate the gain provided by a (re)transmission of a data unit only based on the transmission history of its dependent data units, get stuck once the ACK of an ancestor data unit is lost or delayed. In that case, the scheduler can not afford to wait before retransmitting the ancestor data unit in question, because the benefit to expect from the transmission of other descendant data units remains negligible as long as the ancestor has a good chance to be lost. However, such premature retransmission impairs the overall RD performance if the ACK was just delayed. By taking the arrival likelihood into account, PG-AL prevents the expected benefit for descendant data units to drop too fast once an ancestor feedback is delayed. As a consequence, PG-AL allows transmissions of these descendants data units, which in turn permit a longer wait to get more knowledge about the ancestor status. Here, it is worth noting that this delayed ACK problem becomes irrelevant when several independent groups of data units are considered simultaneously for transmission. Indeed, in that case, the scheduler can always postpone the retransmission of a problematic ancestor data unit, by first transmitting a data with larger expected benefit in another group.

By comparing the plots in Figures 6 (a), (b) and (c), we observe that, for all scheduling algorithms, the streaming performance improves as W_τ increases, but progressively saturates once W_τ gets larger than the GOP size (i.e., $W_\tau > 20$). We explain such behavior as follows. A pre-fetching window smaller than the GOP size is not recommended, because it prevents the scheduler to take all dependent frames into account when selecting the transmission strategy for a given frame. Further analysis of Figures 6 (b) and (c) also reveals that patient greedy algorithms gets closer to the RD optimal performance as the pre-fetch window W_τ increases beyond the GOP size. In particular, we observe that PG-AL achieves performance that are nearly equivalent to ARC-RaDiO for $W_\tau = 30$. We conclude that patient greedy algorithms are able to achieve close to RD optimal performance as long as the pre-fetch delay (i) gets larger than 2 or 3 RTTs (see Section V-B) and (ii) is large enough to schedule simultaneously more than one group of interdependent data units.

Finally, Figure 7 considers a channel with reliable feedback, i.e., there are no loss on the reverse path, and confirms the above conclusions. Again, we observe that PG-AL brings a benefit over PG, and achieves performance that are much closer to the RD bounds than the one obtained based on the conventional greedy algorithm.



(a) $\varepsilon_F = 0.1$ (b) $\varepsilon_F = 0.2$

Fig. 7. PSNR quality versus bitrate, when streaming the QCIF Foreman video sequence. The channel conditions are defined by $\mu_F = \mu_B = 100ms$, and $\varepsilon_B = 0$. The pre-fetch window is set to $W_\tau = 20$.

In summary, the simulation results leads to three main conclusions, as follows:

- First, the proposed patient greedy algorithm significantly outperforms conventional greedy mechanisms.
- Second, for sufficiently large values of the authorized pre-fetch delay, patient greedy algorithms reach performances that are competitive compared to the ones achieved by the computationally complex ARC-RaDiO scheme. Hence, the patient greedy mechanism appears to be able to capture the essence of RaDiO algorithms, while preserving the simplicity of a greedy approach.
- Third, the patient greedy algorithm even improves by estimating the *a priori* probabilities of arrival for data units. In other words, PG-AL performs better than PG. We have observed in our simulations that the benefit of PG-AL over PG is only significant when only one or two groups of interdependent data (GOD) are considered simultaneously for transmission at any given time. In that case, the PG scheduler might not have the opportunity to postpone the retransmission of a data unit, because all its descendants are useless until the ancestor status is fixed, and because, beyond these descendants, there are no other data units to transmit. In contrast, when the pre-fetching window contains a large number of GODs, it is always possible for the scheduler to switch between GODs so as to wait for a sufficiently reliable information about the reception of ancestors, before triggering its retransmission. This second scenario is the one encountered for articial data in Section V-B.

VI. CONCLUSIONS

This paper presents a low complexity, rate adaptive, scheduling algorithm, which offers close to optimal rate-distortion performances in streaming packetized media data. We study the sub-optimality of popular greedy schedulers, which are penalized by premature retransmissions of media packets. Therefore, we propose to delay the packet retransmission decisions, and present a patient greedy (PG) scheduling algorithm to alleviate the distortion penalty due to greedy scheduling, while preserving the low computational complexity of these approaches. Patient greedy algorithms provide a rate-distortion performance that is close to optimal, as long as the pre-fetch delay (i) is larger than a few average round-trip times on the communication channel and (ii) is sufficiently large so that the scheduler can consider simultaneously at least one entire group of data units for transmission at any given time instant. If the pre-fetch delay becomes small, the patient greedy however degenerates in a conventional greedy scheduling solution. Simulation results show that, when streaming packetized video content, the patient greedy algorithm typically provides a 2 dB improvement in end-to-end rate-distortion performance relative to conventional greedy schedulers, with a comparable computational complexity. Patient greedy scheduling algorithms therefore provide low complexity and efficient solution for practical streaming scenarios, mostly for on-demand media applications.

APPENDIX A
ERROR CONCEALMENT

Sections II-B, III-A, and IV reveal that the sensitivity S_l^t estimated at time t for data unit l plays a key role for both greedy and RaDiO-based sheduling mechanisms. Formally, the sensitivity S_l^t is the amount by which the quality is expected to increase (or equivalently the distortion is expected to decrease) if data unit l reaches the receiver on-time, given the transmission decisions made for l and other data units at time t.

When the incrementally additive distortion model strictly holds within a group of interdependent data units, the amount by which the distortion is decreased (the quality is increased) if l is decoded, compared to the distortion if only the ancestors of l are decoded, is denoted ΔD_l . In that case, the sensitivity S_l^t is defined by [5]

$$S_l^t = \sum_{l' \succeq l} \left(\Delta D_{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'') \right) , \tag{23}$$

where $p_c^t(i)$ denotes the probability that data unit i reaches the client in time, given the transmission decisions made for data unit i at time t. Formally, in the greedy case, $p_c^t(i) = p_c^t(i \mid \varkappa_i^t)$. In the RaDiO case, we have $p_c^t(i) = 1 - \epsilon(\pi_i^t)$.

In presence of concealment, the incrementally additive model does not hold anymore. The decrease in distortion if data unit l is decoded depends on whether cousin or sibling data units have been decoded or not [5]. Here, reference and sibling data units of l denote the data units that are used to conceal l or that refer to l for their own concealment. The purpose of this appendix is to explain how the sensitivity S_l^t is computed in presence of error concealment. We follow the approach introduced by Chakareski and Girod in [22]. Let Γ_l denote the set of data units that are considered to conceal data unit l. Note that $l \in \Gamma_l$ and that the decoder is supposed to select the most recent decodable frame in Γ_l to conceal l. Let also ΔD_l^j denote the reduction in distortion if data unit l is concealed based on data unit $j \in \Gamma_l$. We then introduce $P^t[i \leftarrow j \mid j]$ to denote the probability, estimated at time t, that data unit i is concealed based on data unit j, knowing that j has been decoded correctly. If j has been decoded correctly, the only condition for i to be concealed by j is that no data unit in Γ_i that is more recent than j has also been correctly decoded. Formally, we denote $\chi(j,i)$ to be the set of data units that should not be received to force concealment of data unit i by data unit j, and we have

$$P^t[i \leftarrow j \mid j] = \prod_{k \in \chi(j,i)} (1 - p_c^t(k)) \ . \tag{24}$$

Based on the above definitions, the sensitivity S_l^t in presence of concealment is written

$$\begin{aligned} S_l^t &= \sum_{l' \succeq l} \sum_{i: l' \in \Gamma_i} \Delta D_i^{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'') \prod_{k \in \chi(l',i)} (1 - p_c^t(k)) \\ &\quad - \sum_{i: l \in \Gamma_i} \sum_{j \in \Gamma_i : l \in \chi(j,i)} \Delta D_i^j \prod_{j' \preceq j} p_c^t(j') \prod_{k \in \chi(j,i), k \neq l} (1 - p_c^t(k)) \ . \end{aligned} \tag{25}$$

The first term in Equation (25) corresponds to the gain in quality resulting from a correct reception of data unit l, while the second term, i.e., the subtraction, reflects the fact that error concealment reduces the impact of the absence of data unit l on the reconstructed quality.

We now develop Equation (25) for the case of interest in section V-C. There, the Foreman video sequence is encoded as a succession of Group of Pictures (GOP). Each GOP contains $N_g = 20$ frames, consisting of one I frame followed by 19 P frames. The scheduler approximates the decoder concealment strategy by considering that each frame is concealed by repeating the most recent decodable frame among its $N_c = 30$ immediately preceding frames. In that case, we denote ΔD_i^j to be the Y-PSNR of frame i when it is approximated by the correctly decoded frame j, and Equation (25) becomes

$$S_l^t = \left(\sum_{l' \succeq l} \sum_{i=l'}^{l'+N_c} \Delta D_i^{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'') \prod_{k \in \chi(l',i)} (1 - p_c^t(k)) \right) - C_l^t \ , \tag{26}$$

where

$$C_l^t = \begin{cases} \sum_{i=l}^{l-1+N_c} \Delta D_i^{l-1} \prod_{l' \preceq l-1} p_c^t(l') \prod_{k \in \chi(l-1,i)} (1 - p_c^t(k)) & \text{if l is an INTER frame} \\ \sum_{i=l}^{l-1+N_c} \sum_{j \in \Gamma_i, j<l} \Delta D_i^j \prod_{l' \preceq j} p_c^t(l') \prod_{k \in \chi(j,i), k \neq l} (1 - p_c^t(k)) & \text{if l is an INTRA frame .} \end{cases}$$

As explained earlier, the set $\chi(l,i)$ contains the data units that should not be received on time in order to force concealment of data unit i by data unit l. Formally, $\chi(l,i)$ is empty if $i \leq l$. When $i > l$, $\chi(l,i)$ contains $l+1$ and all Intra frames j so that $l < j \leq i$.

REFERENCES

[1] Girod, B. and Chakareski, J. and Kalman, M. and Liang, Y. J. and Setton, E. and and Zhang, R., "Advances in network-adaptive video streaming," in *Proc. 2002 Tyrrhenian International Workshop on Digital Communications (IWDC 2002)*, Capri, Italy, September 2002, pp. pp. 1–8.

[2] Zhi-Li Zhang, Srihari Nelakuditi, Rahul Aggarwal, and Rose P. Tsang, "Efficient selective frame discard algorithms for stored video delivery across resource constrained networks," in *Proceedings of IEEE INFOCOM*, 1999, pp. 472–479.

[3] Krasic, C. and Walpole, J. and Feng, W.-C., "Quality-adpative media streaming by priority drop," in *Proceedings of the ACM NOSSDAV*, June 2003.

[4] P. de Cuetos and K.W. Ross, "Adaptive rate control for streaming stored rine-grained scalable video," in *NOSSDAV '02: Proceedings of the 12th international workshop on Network and Operating Systems Support for Digital Audio and Video*, New York, NY, USA, 2002, pp. 3–12, ACM Press.

[5] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, May 2005, accepted for publication.

[6] M. Röder, J. Cardinal, and R. Hamzaoui, "On the complexity of rate-distortion optimal streaming of packetized media," in *Proc. IEEE Data Compression Conference*, Snowbird, UT, March 2004, IEEE Computer Society, pp. 192–201.

[7] M. Röder, J. Cardinal, and R. Hamzaoui, "Branch and bound algorithms for rate-distortion optimized media streaming," *IEEE Trans. Multimedia*, 2005, in press.

[8] J. Chakareski and P. Frossard, "Rate-distortion optimized packet scheduling over bottleneck links," in *Proc. Int'l Conf. Multimedia and Exhibition*, Amsterdam, The Netherlands, July 2005, IEEE.

[9] Danjue Li, Gene Cheung, Chen-Nee Chuah, and S.J. Ben Yoo, "Joint server/peer receiver-driven rate-distortion optimized video streaming using asynchronous clocks," in *Proc. IEEE International Conference on Image Processing (ICIP)*, Singapore, October 2004.

[10] D. Tian, X. Li, G. Al-Regib, Y. Altunbasak, and J.R. Jakson, "Optimal packet scheduling for wireless video streaming with error-prone feedback," in *IEEE WCNC*, Atlanta, March 2004.

[11] Z. Miao and A. Ortega, "Optimal scheduling for the streaming of scalable media," in *Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, October 2000.

[12] Sehgal, A. and Jagmohan, A. and Verscheure, O. and Frossard, P., "Fast distortion-buffer optimized streaming of multimedia," in *Proc. IEEE International Conference on Image Processing (ICIP)*, September 2005.

[13] A.C. Begen, Y. Altunbasak, and M.A. Begen, "Rate-distortion optimized on-demand media streaming with server diversity," in *Proc. IEEE International Conference on Image Processing (ICIP)*, Barcelona, Spain, September 2003.

[14] J. Chakareski, P.A. Chou, and B. Girod, "Rate-distortion optimized streaming from the edge of the network," in *IEEE Workshop on Multimedia Signal Processing (MMSP)*, St. Thomas, US Virgin Islands, December 2002.

[15] J. Chakareski and B. Girod, "Computing rate-distortion optimized policies for streaming media with rich acknowledgements," in *Proc. IEEE Data Compression Conference*, Snowbird, UT, April 2004.

[16] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoustics Speech and Signal Processing*, vol. 36, no. 1, pp. 1445–1453, September 1988.

[17] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *IEEE Trans. Image Processing*, vol. 2, no. 2, pp. 160–175, April 1993.

[18] J. Chakareski, J. Apostolopoulos, and B. Girod, "Low-Complexity Rate-Distortion Optimized Video Streaming," in *IEEE International Conference on Image Processing (ICIP)*, Singapore, October 2004.

[19] A. Ortega, "Optimal bit allocation under multiple rate constraints," in *IEEE Data Compression Conference (DCC)*, Snowbird, Utah, USA, April 1996, pp. 349–358.

[20] C.-Y. Hsu, A. Ortega, and M. Khansar, "Rate control for robust video transmission over burst-error wireless channel," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 5, pp. 756–773, May 1999.

[21] C.-L. Chang, S. Han, and B. Girod, "Rate-distortion optimized streaming for 3-D wavelet video," in *Proc. IEEE International Conference on Image Processing (ICIP)*, Singapore, October 2004.

[22] J. Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *IEEE Data Compression Conference (DCC)*, Snowbird, Utah, USA, March 2003.