The Virtue of Patience in Low-Complexity Scheduling of Packetized Media With Feedback

Christophe De Vleeschouwer, Jacob Chakareski, and Pascal Frossard, *Senior Member, IEEE*

***Abstract*—We consider streaming pre-encoded and packetized media over best-effort networks in the presence of acknowledgment feedbacks. We first review a rate-distortion (RD) optimization framework that can be employed in such scenarios. As part of the framework, a scheduling algorithm selects the data to send over the network at any given time, so as to minimize the end-to-end distortion, given an estimate of channel resources and a history of previous transmissions and received acknowledgements. In practice, a greedy scheduling strategy is often considered to limit the solution search space, and reduce the computational complexity associated to the RD optimization framework. Our work observes that popular greedy schedulers are strongly penalized by early retransmissions. Therefore, we propose a scheduling algorithm that avoids premature retransmissions, while preserving the low computational complexity aspect of the greedy paradigm. Such a scheduling strategy maintains close to optimal RD performance when adapting to network bandwidth fluctuations. Our experimental results demonstrate that the proposed patient greedy scheduler provides a reduction of up to 50% in transmission rate relative to conventional greedy approaches, and that it brings up to 2 dB of quality improvement in scheduling classical MPEG-based packet video streams.**

***Index Terms*—Audio coding, channel coding, error control, greedy packet scheduling, Internet, Markov processes, multimedia communicationoptimal control, protocols, rate-distortion optimization, video coding, video streaming.**

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, in order to become a reality a 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 [1], which can be roughly classified into four categories: 1) versioning, with several encodings of the same source at different rates and different error protection levels, but this solution often suffers from coarse granularity; 2) transcoding within the network, which however often results in computationally complex solutions; 3) scalable coding of the media stream; and 4) efficient data packet selection and scheduling, which minimize the end-to-end distortion under stringent timing and channel constraints.

TABLE I
SCHEDULER ACRONYMS

Acronym	Definition
G	Conventional greedy scheduler (see Section III-A)
PG	Patient greedy scheduler (see Section IV-B)
PG-AL	PG + arrival likelihoods (see Section IV-C)
RaDiO	Rate distortion optimized scheduler (see Section II-B)
ARC-RaDiO (RD)	RaDiO with adaptive rate control (see Section II-C)

In this paper, we examine solutions from the last category that address the scenario of sender-driven streaming of packetized media with acknowledgements feedback. Many of these works, such as [2]–[5], consider the relative importance of the media data units when performing packet scheduling decisions. However, they typically propose greedy algorithms for packet selection and rate allocation, thereby ignoring future expected transmissions for dependent media packets, which in turn results into their suboptimal rate-distortion (RD) performance. This is often done in order to reduce the complexity of the original optimization problem and make it tractable. Another related work is [11], which analyzes the impact of a very narrow optimization horizon on the scheduling algorithm and its RD performance. As such, it provides a complementary study to our work. The study in [11] is performed in the context of a reliable lossless channel with fluctuating bandwidth. Due to the limited search horizon, the proposed scheduling algorithm is greedy in nature. Hence, to deal with this drawback, the authors in [11] augment the expected distortion function with the buffer occupancy level, such as to enforce maintaining a certain predefined buffer occupancy level. Our work does not focus on a very limited horizon. Conversely to [11], it envisions media streaming over channels that are subject to packet losses and random delays.

A scheduling framework that jointly considers media units' dependencies their history of previous transmissions, and their expectation of future transmissions with the goal of rate-distortion optimality has been formalized in [6] and [7]. Solutions to the associated RD problem under investigation have been proposed in [6]–[9]. However, these solutions exhibit high computational complexity, which makes their use unfeasible in many practical scenarios. In addition, they also do not provide any guarantees on the sending rate trace, and thus may not be able to respect instantaneous network bandwidth fluctuations.

The goal of the present paper is to propose a streaming solution that is able to adapt to instantaneous rate constraints, with

Manuscript received August 24, 2005; revised June 8, 2006. This work was supported by the Swiss National Science Foundation (NSF), under Grant PP-002-68737, and by the Belgian NSF. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ching-Yung Lin.

C. De Vleeschouwer is with the Laboratoire de Télécommunications, Université Catholique de Louvain, Louvain-la-Neuve, Belgium (e-mail: devlees@tele.ucl.ac.be).

J. Chakareski was with the Signal Processing Institute—LTS4, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. He is now with Layered Media, Inc., Rochelle Park, NJ 07662 USA (e-mail: io@jakov.org).

P. Frossard is with the Signal Processing Institute—LTS4, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland (e-mail: pascal.frossard@epfl.ch).

Digital Object Identifier 10.1109/TMM.2006.886283

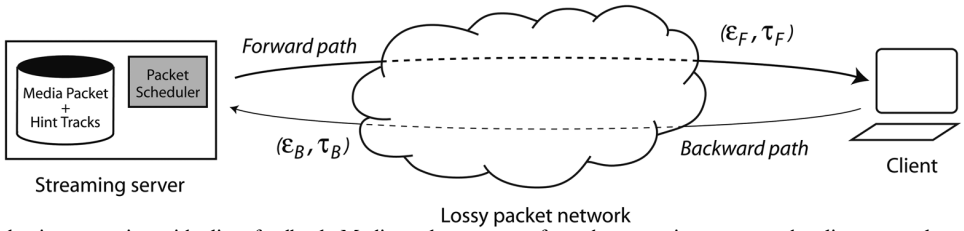

Fig. 1. Sender-driven adaptive streaming with client feedback. Media packets are sent from the streaming server to the client over a lossy packet network, whose channel paths are characterized by random delays τ_F and τ_B, and packet loss probabilities ε_F and ε_B.

low computational complexity, and yet to provide a close to optimal performance from an end-to-end distortion perspective. To this end, we first review an extension of the RaDiO framework, called ARC-RaDiO, which is able to adapt to instantaneous rate constraints, with the introduction of a sending buffer. This scheduler, initially proposed in [10], offers an optimized RD performance, but at the price of high computational complexity, even larger than the one of the original RaDiO algorithm.

Our main contribution consists in deriving computationally tractable, while still close-to-optimal, scheduling solutions for streaming scenarios where the server has to match the rate variations experienced on the network connection.

Therefore, our paper defines a new patient greedy (PG) scheduler, which maintains the acceptable computational complexity of greedy approaches, but improves 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 earlier transmissions are likely to arrive at the sender in the near 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 to compute the impact of such a decision on the expected RD performance. The resulting PG scheduler outperforms conventional greedy strategies without increasing their computational complexity, and provides in addition a close to optimal RD performance.

The rest of the paper is organized as follows. Section II presents the streaming framework considered in this paper, and formalizes the RD 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 suboptimality of conventional greedy approaches is also examined in detail. In Section IV, we present our novel PG packet scheduler and, we analyze its performance via simulations in Section V. Finally, we provide concluding remarks in Section VI. For clarity purposes, Table I summarizes the acronyms that are employed to denote the various scheduling algorithms under investigation in this paper.

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 [7]. 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 be 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 lth 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 correctly decoded, compared to the distortion induced when only the ancestors of l are decoded.

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 into more than one packet (i.e., retransmissions are possible), but we assume that a packet contains one and only one data unit. In practice, the scheduler works at the application level, and does not care about congestion control issues. It relies on the transport layer to obtain information about packet transmission opportunities. In our work, as in [7], the network forwarding path is modeled as an independent time-invariant packet erasure channel with random delays. This 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 FTT (BTT) random variable, which 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 Fig. 1. As a support to the reader, Table II compiles the main symbols defined throughout the rest of the paper.

B. The RaDiO Framework

One of the most well known works that address the problem of scheduling the transmissions of interdependent media data units is the Rate-Distortion Optimization (RaDiO) framework, introduced originally by Chou and Miao [6], [7], and further

TABLE II
SYMBOL DEFINITION

Symbol	Definition
$\varepsilon_F/\varepsilon_B$	Average forward/backward packet loss probability.
μ_F/μ_B	Average forward/backward trip time.
τ	Prefetching delay (in ms).
W_τ	Prefetching window (in number of frames).
Γ_τ^t	Set of data units whose delivery deadlines lie between t and $t+\tau$.
Ψ_τ^t	Set of PG-eligible data units contained in Γ_τ^t (see Section IV-B).
$l' \prec l$	Partial order ancestor/descendant relationship among L interdependent data units. Data unit l can only be decoded if data unit l' has been decoded.
S_l	Size of the l^{th} data unit.
ΔD_l	Reduction of distortion obtained when decoding the l^{th} data unit compared to the distortion obtained when decoding all of its ancestors.
π_l	N-tuple binary vector defining transmission policy for the l^{th} data unit.
$\vec{\pi}$	Policy vector for L data units.
$D(\vec{\pi})$	Expected distortion associated with $\vec{\pi}$.
$R(\vec{\pi})$	Expected average transmission rate associated with $\vec{\pi}$.
$R_j(\vec{\pi})$	Expected transmission rate at the j^{th} transmission opportunity for a policy vector $\vec{\pi}$.
$\varkappa_l^t$	Transmission history, i.e., instants of previous transmissions, for the l^{th} data unit at time t.
$p(l \mid \varkappa_l^t)$	Estimation at time t of the probability that the l^{th} data unit is received on time at the client given its transmission history $\varkappa_l^t$.
$\beta_l^{t,t'}$	Expected decrease in distortion estimated at current time t for the (re)transmission of the l^{th} data unit at time $t' \geq t$.
P_l	Number of time instances at which the l^{th} data unit has been transmitted in the past.
$p_a^t(k)$	Estimation at time t of the *a priori* probability that the k^{th} data unit reaches the client on time.

studied by Röder *et al.* [8], [9]. The RaDiO approach has laid down the groundwork for recent studies on streaming media from multiple servers [12], [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 have considered scheduling media content over unreliable networks based on RD optimization techniques [2], [4], [5]. In this section, we review the original RaDiO framework, with a special emphasis on its computational complexity.

To compute RD optimal transmission schedules for a GOD, the authors in [6], [7] describe the transmission of L interdependent data units by a policy vector $\vec{\pi} = (\pi_1, \ldots, \pi_L)$, where π_l, for $l \in \{1, \ldots, L\}$, denotes the transmission policy associated with the lth 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 $\{t_{l,0}, t_{l,1}, \ldots, t_{l,N_l-1}\}$, corresponding to the N_l transmission opportunities assigned to data unit l. Hence, $\pi_l = (\pi_l(0), \pi_l(1), \ldots, \pi_l(N_l-1)) \in \{0,1\}^{N_l}$, where $\pi_l(i) = 1$ means that data unit l should be sent at opportunity i if it is not yet acknowledged by then. The expected error $\epsilon(\pi_l)$ associated with policy π_l defines the probability that data unit l does not reach its destination before its delivery deadline $t_{d,l}$ as

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

where it is assumed that the individual packet transmissions are independent. Furthermore, the expected cost $\rho(\pi_l)$ is defined as the expected number of data unit (re)transmissions

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

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 Lagrange cost $J_\lambda(\pi_l) = \epsilon(\pi_l) + \lambda\rho(\pi_l)$, for some $\lambda > 0$. They can be computed with worst-case complexity of $O\left(N_l 2^{N_l}\right)$ [8], which grows exponentially in the number of transmission opportunities.

Regarding the GOD, Chou and Miao [7] define the expected transmission rate $R(\vec{\pi})$ and distortion $D(\vec{\pi})$ for a given policy vector $\vec{\pi}$ as

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

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

where D_0 denotes the distortion when no data unit has been received on time, S_l is the size of data unit l in bytes, and ΔD_l is its importance. A convex-hull RD optimal policy vector is then defined as a vector that minimizes the Lagrange 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 $\lambda > 0$ is proposed in [7]. This solution, denoted Iterative Sensitivity Adjustment (ISA), relies on the cost-error optimal policies defined for a single data unit, and is based on an iterative descent algorithm that minimizes $J_\lambda(\vec{\pi}) = J_\lambda(\pi_1, \ldots, \pi_L)$ one policy at a time, while keeping the other policies fixed. Its complexity grows linearly with the number L of dependent data units. 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.* [8] have proposed a branch and bound algorithm to compute a global optimum. However its complexity grows exponen-

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

tially both in the number of transmission opportunities and the number of data units, which is certainly too constraining for an on-line application [8].

Under its original formulation, the RaDiO framework presented above often leads to an 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 a streaming session at any given time, we follow the authors in [7] 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 deadlines lie in the sliding window $[t, t+\tau]$ are given an opportunity to be transmitted. The authors in [7] then propose to select the RD optimal policy vector corresponding to an average rate constraint imposed by the channel. Given an average target rate, a bisection search algorithm [16], [17] adjusts λ in (5), to select the policy vector that: 1) is convex-hull RD optimal and 2) has the largest rate among the optimal policy vectors that respect the rate constraint.

The computational complexity associated with 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 deadlines lie in the sliding window $[t, t+\tau]$. This is done with a total worst-case complexity of $O(LN2^N)$, N being the number of transmission opportunities for each data unit [8]. For a given Lagrange multiplier λ, the RD 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 a complexity of $O(LS)$. 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 $n_{i\lambda}$ times, until either the target rate is achieved or the λ interval becomes smaller than a pre-defined threshold. Our simulations have revealed that ten iterations are sufficient to reach a λ value for which the computed optimal transmission policies correspond to the average target transmission rate. Therefore, the overall complexity of the whole procedure is $O\left(LN2^N + n_{i\lambda} n_{i\vec{\pi}} LS\right)$. 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 tractable. Note that an in-depth analysis of the computational complexity of a RaDiO packet scheduler is presented 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 requirements for channel resources that are incompatible with the network policy in place. 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 rate. A buffer absorbing the mismatch would introduce unnecessary delays in the transmission thereby resulting in suboptimality.

This important observation has motivated the authors in [6], [7] to propose a simplification of the original RaDiO scheduling algorithm that only approximates the RD optimal solution, but which is able to adapt to the dynamic channel rate variations. The λ parameter is adjusted so that exactly one data unit is selected for transmission at each opportunity. Note that the equations provided in [7] reveal that this particular case is in essence equivalent to a conventional greedy scheduling solution that expectedly stays suboptimal from an RD perspective (as explained later in Section III-B).

In order to address this limitation, we now introduce a variation of the original RaDiO framework [6], [7], called ARC RaDiO that is able to meet instantaneous channel bandwidth constraints while preserving RD 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 [10].

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[2] with a period equal to T. 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 deadlines $t_{d,l}$ lies in the transmission window $[t_i, t_i + \tau]$ are given the opportunity to be transmitted. Similar to Section II-B, let $\pi(\vec{t_i}) = (\pi_1, \ldots, \pi_{L(t_i)})$ 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 are limited to the set of time instances defined by $\{t_i, t_i + T, \ldots, t_i + (N-1)T)\}$, 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 [7] and [10]).

To formalize the buffer management process, ARC-RaDiO extends the notion of expected cost by incorporating the time at which data units are transmitted. Hence, $\rho(\pi_l(j))$ denotes the expected cost under policy π_l at the jth transmission opportunity, i.e., at time $t_i + jT$. At the GOD level, we define

[2]The sampling period 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.

$R_j(\pi(\vec{t}_i))$ to be the expected transmission rate at the jth transmission opportunity, for the policy defined at current time t_i. For $j = 0, \ldots, N-1$

$$R_j(\pi(\vec{t}_i)) = \sum_{l=1}^{L(t_i)} S_l \rho(\pi_l(j)). \tag{6}$$

The ARC-RaDiO optimal policy vector at current time t_i is denoted $\pi(\vec{t}_i)^*$, and is the one that minimizes the expected distortion computed on $L(t_i)$ data units based on (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 rate R_c. Note that the draining rate can be made dynamic and varying over time, without significantly changing the following development. Formally, we have

$$\pi(\vec{t}_i)^* = \underset{\pi(\vec{t}_i)}{\operatorname{argmin}} D(\pi(\vec{t}_i))$$
$$\text{s.t.} \sum_{j=0}^{k} R_j(\pi(\vec{t}_i)) \leq kTR_c + BS(t_i),$$
$$\text{for } k = 0, \ldots, N-1. \tag{7}$$

In particular, we note in (7) that the constraint corresponding to $k = 0$ is a deterministic constraint. That is because the transmissions of data units at current time t_i that are recommended by $\pi(\vec{t}_i)$ do not depend on the (non-)reception of feedbacks in the future. They are effective and define the actual rate at the 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, we omit the dependency on the current time t_i for $\pi(\vec{t}_i)$ and all related variables. As in [10], the constrained optimization is reformulated as an unconstrained problem using Lagrange multipliers. Hence, we look for the policy vector that minimizes the Lagrangian

$$J_{\vec{\lambda}}(\vec{\pi}) = D(\vec{\pi}) + \sum_{k=0}^{N-1} \lambda_k \sum_{j=0}^{k} R_j(\vec{\pi}) \tag{8}$$

for a vector of positive Lagrange multipliers $\vec{\lambda} = [\lambda_0, \ldots, \lambda_{N-1}]$. By rearranging (8) and defining $\lambda'_k = \sum_{j=k}^{N-1} \lambda_j$, for $k = 0, \ldots, N-1$, the Lagrangian in (8) becomes

$$J_{\vec{\lambda}'}(\vec{\pi}) = D(\vec{\pi}) + \sum_{k=0}^{N-1} \lambda'_k R_k(\vec{\pi}). \tag{9}$$

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 $J_{\vec{\lambda}'}(\pi_1, \ldots, \pi_L)$ one policy at a time, keeping the other policies fixed [10]. The algorithm is similar to the algorithm presented in [7] 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, [6], [7] demonstrate that the RD optimal transmission policy for a single data unit, given other data unit policies, is directly provided by the expected error-cost convex-hull that is computed once and for all, with $O(N2^N)$ complexity. In contrast, because ARC-RaDiO considers a vector of N λ'_k values, the convex-hull becomes an N-dimensional surface so that the mapping between an arbitrary vector of Lagrangian multipliers and its corresponding optimal policy can not be handled efficiently anymore. For this reason, ARC-RaDiO is forced to perform an exhaustive search over the entire set of policies each time the iterative algorithm has to optimize a 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 $O\left(n_{i\vec{\pi}} LN2^N + n_{i\vec{\pi}} LS\right)$. In comparison with the RaDiO framework, we note that the dominating factor $LN2^N$ 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 a selection for the set of Lagrange multipliers that solves the initial problem formulated in (7), and thus results in a constrained transmission rate close 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 [10]. Let $n_{i\vec{\lambda}'}$ denote the number of iterations that is needed to converge to an 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 at least N times 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 $O\left(n_{i\vec{\lambda}'} n_{i\pi} \left(LN2^N + LS\right)\right)$, which is certainly greater than the complexity of the original RaDiO framework.

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

III. Towards Low-Complexity Scheduling

A. Greedy Strategy

As discussed earlier, several works have proposed to control streaming systems based on RD optimization techniques by formalizing the packet scheduling decisions as a partially observable Markov decision process [2], [4], [5], [7]. Furthermore, in order to limit the associated computational complexity and/or to match the instantaneous rate imposed by the network, these works generally end-up recommending the use of heuristic greedy scheduling mechanisms that transmit, at any given time, the data unit that maximizes the expected decrease in distortion

per unit of transmitted rate. In the rest of the paper, we denote such schemes as conventional greedy scheduling and in general they represent simplified solutions of RD optimal packet scheduling problems. For example, [7] proposes to adjust the λ parameter in (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 [2] and [21].

More formally, the greedy (G) scheduling mechanisms proposed by all these works can be summarized as follows. Let t and τ denote respectively 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. Let the set of data units whose delivery deadlines lie between t and $t+\tau$ be denoted as 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 lth data unit at time t. Specifically, $\varkappa_l^t = \left\{t_l^1, t_l^2, \ldots, t_l^{P_l}\right\}$ defines the P_l time instants at which the lth data unit has been transmitted in the past. We now estimate the probability $p\left(l \mid \varkappa_l^t\right)$ for the lth data unit to be received on time at the client, knowing its transmission history $\varkappa_l^t$. When an acknowledgment has been received for data unit l, $p\left(l \mid \varkappa_l^t\right)$ is obviously equal to 1. In the 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. Therefore, assuming independent packet transmissions as in [6] and [7], we write

$$p\left(l \mid \varkappa_l^t\right) = 1 - \prod_{i \le P_l} P\left\{\mathrm{FTT}_i^l > t_{d,l} - t_l^i \mid \mathrm{RTT}_i^l > t - t_l^i\right\} \quad (10)$$

where FTT_i^l and RTT_i^l denote, respectively, the forward- and round-trip time random variables associated with the ith (re)transmission of the lth data unit. We now estimate the reduction in distortion β_l^t that can be expected at time t from an additional transmission of the lth data unit. Taking into account the dependencies among the data units, we have

$$\beta_l^t = \left[p\left(l \mid \varkappa_l^t \cup \{t\}\right) - p\left(l \mid \varkappa_l^t\right)\right] \times \sum_{l' \succeq l} \left(\Delta D_{l'} \prod_{l' \preceq l', l'' \neq l} p\left(l'' \mid \varkappa_{l''}^t\right)\right). \quad (11)$$

The sum in (11) reflects the fact that the reception of the lth data unit is beneficial for l but also for all of its descendants. The product in (11) expresses the fact that correct decoding of data unit l' is subject to on time reception of all of its ancestors. At current time t, the greedy approach (re)transmits the data unit, denoted $l^G(t)$, which maximizes the expected gain in distortion per unit of rate. Therefore, we have

$$l^G(t) = \underset{l \in \Gamma_\tau^t}{\operatorname{argmax}} \frac{\beta_l^t}{S_l}. \quad (12)$$

In terms of complexity, the greedy algorithm is dominated by the computations involved in (11). 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 units, while L accounts for the search for the data unit with the highest RD ratio. Here, we neglect the complexity associated with (10), 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 zero or one. Since nothing comes for free, greedy strategies have however to pay a penalty in terms of RD performance.

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

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	-

B. Complexity Versus Optimality Tradeoff

It is quite obvious that limiting the search space can only penalize the performance of the resulting solution. Such a distortion penalty is however difficult to appreciate. In order to further understand the suboptimality induced 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 mechanisms, defined, respectively, in Sections II-C and III-A. The behaviors of these schedulers are compared in a toy example, which 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 five hierarchical layers, defined such that $S_{l+1} = S_l = 50$ bits and $\Delta D_{l+1} = \Delta D_l/2, \forall\, l \le 5$. The frames are encoded at a rate of 20 fps. The maximal pre-fetching delay τ is set to 1 s. The channel delay probabilitiy density functions (pdfs) are modeled as shifted exponentials with mean $\mu_F = \mu_B = 180$ ms. The channel loss rates are defined by $\varepsilon_F = 0.2$, and $\varepsilon_B = 0$, and the channel transmission rate is set to 6500 bits/s (= 325 bits/frame). Table III compares the statistics of the greedy and the RD optimal algorithms, and presents: 1) the percentage of data units that have reached the client on time; 2) the average number of transmissions per data unit; and 3) 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 may result in significantly suboptimal RD tradeoffs. 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 III, we conclude that a lot of these retransmissions could

be avoided if the sender was more patient in triggering retransmissions: the RD optimal scheduling almost never retransmits 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 III 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 an immediate retransmission of data units in case a delayed retransmission is likely to bring a benefit in the RD sense. The resulting PG scheduling solution still preserves the simplicity offered by the conventional greedy scheduler since the search space stays limited. Moreover, an improved RD performance is achieved by considering a longer time horizon for the optimization, as it enables accounting for prospective future arrivals of acknowledgements.

IV. Patient Greedy Scheduling

Based on the analysis presented in Section III-B, we now propose a scheduling algorithm that prevents premature retransmissions. In brief, Section IV-A analyzes the impact of postponing a retransmission. Based on this analysis, Sections IV-B and IV-C define, respectively, two versions of our proposed PG algorithm, which differ only in their on-line estimation of the probabilities of arrival of data units at the receiver. Finally, Section IV-D discusses 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 PG scheduling algorithm, we now formally examine the impact of a delayed retransmission of a data unit on the RD formulation of the problem under investigation. Let $\beta_l^{t,t'}$ denote the expected decrease in distortion estimated at current time t for the (re)transmission of the lth data unit at time $t' \geq t$. Following (11), we write

$$\beta_l^{t,t'} = \left[p\left(l \mid \varkappa_l^t \cup \{t'\}\right) - p\left(l \mid \varkappa_l^t\right)\right] \times \sum_{l' \geq l} \left(\Delta D_{l'} \prod_{l'' \leq l', l'' \neq l} p\left(l'' \mid \varkappa_{l''}^t\right) \right). \quad (13)$$

On the right-hand side of (13), only the term $p\left(l \mid \varkappa_l^t \cup \{t'\}\right)$ depends on t'. Based on (10), and given the transmission history $\left\{t_l^i\right\}_{i \leq P_l}$, in the absence of an ACK packet for data unit l by time t, we have

$$p\left(l \mid \varkappa_l^t \cup \{t'\}\right) = 1 - P\left\{\mathrm{FTT}_{P_l+1}^l > t_{d,l} - t'\right\} \times \prod_{i \leq P_l} P\left\{\mathrm{FTT}_i^l > t_{d,l} - t_l^i \mid \mathrm{RTT}_i^l > t - t_l^i\right\} \quad (14)$$

which shows that $p\left(l \mid \varkappa_l^t \cup \{t'\}\right)$, and consequently the benefit of reduction in distortion $\beta_l^{t,t'}$, decreases as t' increases. Furthermore, because an ACK might be received between t and t', the expected cost in transmission rate associated with 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 lth data unit at time $t' \geq t$. Given the transmission history $\left\{t_l^i\right\}_{i \leq P_l}$ of data unit l, we have

$$\zeta_l^{t,t'} = S_l \prod_{i \leq P_l} P\left\{\mathrm{RTT}_i^l > t' - t_l^i \mid \mathrm{RTT}_i^l > t - t_l^i\right\}. \quad (15)$$

B. PG: The Patient Greedy Algorithm

Based on Section IV-A, we know that postponing the retransmission of the l^{th}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 a Lagrange factor $\lambda(t)$, which balances the expected gain in rate and the expected loss in quality (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 Lagrange factor $\lambda(t)$, we can decide whether postponing the retransmission from t to t' is likely to bring a global benefit in an RD sense. For the lth data unit, a delayed (re)transmission is beneficial when $\lambda(t)\left(\zeta_l^{t,t} - \zeta_l^{t,t'}\right) > \beta_l^{t,t} - \beta_l^{t,t'}$ or equivalently when

$$-\beta_l^{t,t} + \lambda(t)\zeta_l^{t,t} > -\beta_l^{t,t'} + \lambda(t)\zeta_l^{t,t'}. \quad (16)$$

Based on (16), 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 lth data unit is eligible at time t if

$$t = \underset{t' \in [t, t_{d,l}]}{\operatorname{argmin}} \left(-\beta_l^{t,t'} + \lambda(t)\zeta_l^{t,t'}\right). \quad (17)$$

We now define our proposed patient greedy (PG) scheduling mechanism as a greedy scheduling that is constrained to select the data to transmit from 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 $l^{\mathrm{PG}}(t)$ 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 $\zeta_l^{t,t} = S_l$. Therefore, similarly to (12), we have

$$l^{\mathrm{PG}}(t) = \underset{l \in \Psi_\tau^t}{\operatorname{argmax}} \frac{\beta_l^t}{S_l}. \quad (18)$$

The practical selection of eligible data units in Γ_τ^t is described in Section IV-D.

C. PG-AL: An Extension of PG Exploiting A Priori Probabilities of Arrival

We now explain how PG is refined when some a priori knowledge is available regarding the probabilities of arrival of data units. At current time t, $p\left(k \mid \varkappa_k^t\right)$ from (10) may provide a poor estimate of the actual probability of arrival for data unit k, mainly because possible future transmissions of k are not taken into account. Hence, as a complement to $p\left(k \mid \varkappa_k^t\right)$, we

consider $p_a^t(k)$ to be the estimate at time t of the *a priori* probability that the kth 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 graph. Under this 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 which correspond to the position of data unit k in their respective GODs. In particular, let $k_{\mathrm{GOD}-1}$ denote the latest data unit at location k whose delivery deadline has expired by the current time t. This is the data unit at location k in the previous GOD. Then, the running estimate of the probability $p_a^t(k)$ is updated as $p_a^t(k) \leftarrow \theta p_a^t(k) + (1-\theta) p_a(k_{\mathrm{GOD}-1})$, where θ is the weighting factor and $p_a(k_{\mathrm{GOD}-1})$ denotes the probability of on time arrival for data unit $k_{\mathrm{GOD}-1}$ that can be computed from its transmission history. Finally, note that the argument about employing $p_a^t(k)$ described above implicitly assumes that the scheduling system will assign roughly the same transmission policies to data units at same locations but in different GODs.

Based on the estimates of $p\left(k \mid \varkappa_k^t\right)$ and $p_a^t(k)$, the arrival likelihood (AL) for data unit k is denoted $p_{\mathrm{AL}}^t\left(k \mid \varkappa_k^t\right)$ and is defined as

$$p_{\mathrm{AL}}^t\left(k \mid \varkappa_k^t\right) = \max\left(p\left(k \mid \varkappa_k^t\right), \gamma p_a^t(k)\right). \tag{19}$$

In (19), $\gamma\, p_a^t(k)$ is supposed to provide a lower bound for the probability that k reaches the client on time. In practice, γ results from a tradeoff. 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 $p\left(k \mid \varkappa_k^t\right)$. On the other hand, it should not be too large, so that the lower bound provided by $\gamma\, p_a^t(k)$ stays reliable. In our experiments, we established through empirical data that a value of 0.5 reasonably meets these two requirements. So, γ has been set to 0.5 in our simulations. In addition, the weighting factor θ has been set to 0.75, based on empirical experiments. The benefit expected from a (re)transmission of the lth data unit at time $t' \geq t$, which is estimated based on the arrival likelihood, is denoted $\beta_{l,\mathrm{AL}}^{t,t'}$ and is defined as follows

$$\beta_{l,\mathrm{AL}}^{t,t'} = \left[p\left(l \mid \varkappa_l^t \cup \{t'\}\right) - p\left(l \mid \varkappa_l^t\right)\right] \times \sum_{l' \succeq l} \left(\Delta D_{l'} \prod_{l'' \preceq l', l'' \neq l} p_{\mathrm{AL}}^t\left(l'' \mid \varkappa_{l''}^t\right) \right). \tag{20}$$

The PG algorithm derived from (20) 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: 1) how to estimate the $\lambda(t)$ parameter defined in (16) and (17) and 2) how the eligibility condition defined in (17) is verified in practice. In addition, it also explains that PG and PG-AL have overall complexity that is equivalent to that of greedy scheduling mechanisms.

Regarding the Lagrangian factor $\lambda(t)$, we observe that the rate saved by postponing a retransmission of a data unit is used to (re)transmit one or several other data unit(s). Obviously, it is only beneficial to save 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 to reflect this trade-off, by relating the bit savings and the gain in distortion due to additional data transmissions.

Following the above discussion, we have decided to compute the factor $\lambda(t)$ based on the history of the streaming session. Specifically, $\lambda(t)$ is computed 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. Therefore, $\lambda(t)$ provides a good estimate of the benefit brought by an additional unit of transmitted rate. Following the greedy approach principle, the additional (re)transmissions are then selected as the ones that are expected to bring the largest benefit per unit of rate, from all the prospective transmissions that can be performed at the current time.

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 time interval $[\upsilon_{k-1}, \upsilon_k]$. 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 Lagrange factor based on an exponentially weighted average. We have

$$\lambda\left(\upsilon_k^+\right) = \alpha\lambda_k + (1-\alpha)\lambda\left(\upsilon_k^-\right) \quad \forall\, k > 0. \tag{21}$$

To complete (21), 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 such that the data units within it do not have any ancestor or descendant data units that are outside the group. Typical examples of SGOD are a group of pictures as defined in the MPEG standards, or a frame as defined in the JPEG2000 standard. 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 to be equal to 0.3, but the function $\lambda(t)$ appears to be quite insensitive to the value of α because successive values of λ_k are close to each other.

Given $\lambda(t)$, we now explain how the eligibility condition defined by (17) is checked in practice. Because PG algorithms search for the eligible data unit that has the largest β_l^t/S_l ratio [see (18)], 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 $L' \ll L$. 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 t' values that we examine. 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, rather than 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 PG scheme that examines a smaller number of t' values than the one studied in our simulations. Nonetheless, 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 (13) and (15)]. Therefore, the complexity associated with the eligibility test is $O(L'F)$, which appears to be a negligible increase relative to the $O(LS+L)$ 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 have about the same running time (on a personal computer) as the corresponding implementation of a conventional greedy algorithm.

V. Simulation Results

A. Preliminaries

In this section, we investigate various performance aspects of three scheduling algorithms for streaming packetized media content. The algorithms under examination were described earlier in the paper and are: 1) the RD optimal rate-adaptive scheduling technique denoted as ARC-RaDiO; 2) a conventional greedy scheduler (G); and 3) the PG scheme proposed in this paper. Two types of media content are employed in the experiments: artificial pre-formatted data and actual MPEG-like video packets. The artificial data units correspond to a sequence of identical and temporally equidistant frames, which are decoded independently of each other. Streaming a strictly formatted content provides two major advantages. First, it makes the results easy to reproduce and to compare across different streaming algorithms. Secondly, it facilitates the interpretation and understanding of the underlying scheduling mechanisms, since 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 analyze 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 types of media data. Forward (F) and backward (B) directions on a network path are modeled as independent time-invariant packet erasure channels with random delays (see Section II-A) and constant bandwidth. The pdf's p_F and p_B are modeled as shifted exponentials with mean μ_F (μ_B) and shift $\kappa_F=\mu_F/2$ ($\kappa_B=\mu_B/2$) [7], [21]. In all of our simulations, we consider streaming a sequence of temporally equidistant frames, where the maximum pre-fetching delay τ is defined in terms of the number of frames W_τ contained in a time interval of τ seconds.

To summarize, the results presented in this section demonstrate that the proposed PG algorithms outperform conventional greedy schemes, while reaching close to optimal RD performance. They also show that the optimization horizon over which a scheduler can speculate about future outcomes, defined by the pre-fetching window W_τ, has a significant impact on the streaming performance. For both RD optimal and PG algorithms, the quality improves as the pre-fetching window increases. In contrast, the conventional greedy algorithm is not able to optimally exploit the flexibility offered by a larger pre-fetching window. Our simulations also reveal that taking the arrival likelihood into account in the PG algorithm is beneficial only when a small number of independent GODs can be pre-fetched simultaneously. Otherwise, PG and PG-AL achieve identical RD performance. Finally, we observe that the PG schemes perform closer to the RD optimal algorithms when the pre-fetching delay increases.

B. Pre-Formatted and Artificial Content

Here, we compare the performances of greedy (G), patient greedy (PG), and ARC-RaDiO (RD) algorithms based on streaming of artificial pre-formatted media content[3]. The frame rate of the data is set to 20 fps. 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 lth 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 the absence of any data unit is set to 0. Note that the $R11$ and $R21$ templates are certainly the most realistic ones, as real world media coders generally encode the most important information in the first layers. Finally, the play-out delay of the video application at the receiving client is set to 500 ms.

Fig. 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 RD optimal ARC-RaDiO (RD) algorithms. For each frame, the quality at the receiver is computed as the sum of the increases in quality provided by corresponding decodable data units. The increase in quality associated with a data unit is defined by the distortion template. Fig. 2(a) and (b), respectively, consider the $R11$ and $R21$ distortion templates. In these simulations, the maximum pre-fetching delay τ is set to 1 s, i.e., $W_\tau=20$ frames. We observe from Fig. 2 that PG significantly outperforms G and in addition achieves performance

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

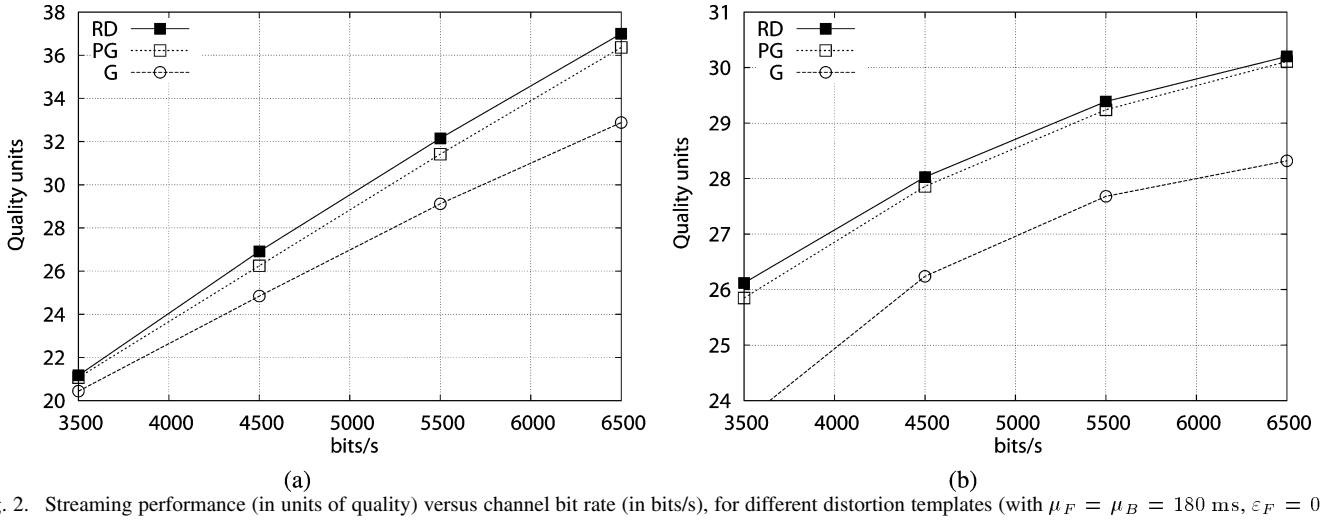

Fig. 2. Streaming performance (in units of quality) versus channel bit rate (in bits/s), for different distortion templates (with $\mu_F = \mu_B = 180$ ms, $\varepsilon_F = 0.2$, and $\varepsilon_B = 0$, and $W_\tau = 20$). (a) R11. (b) R21.

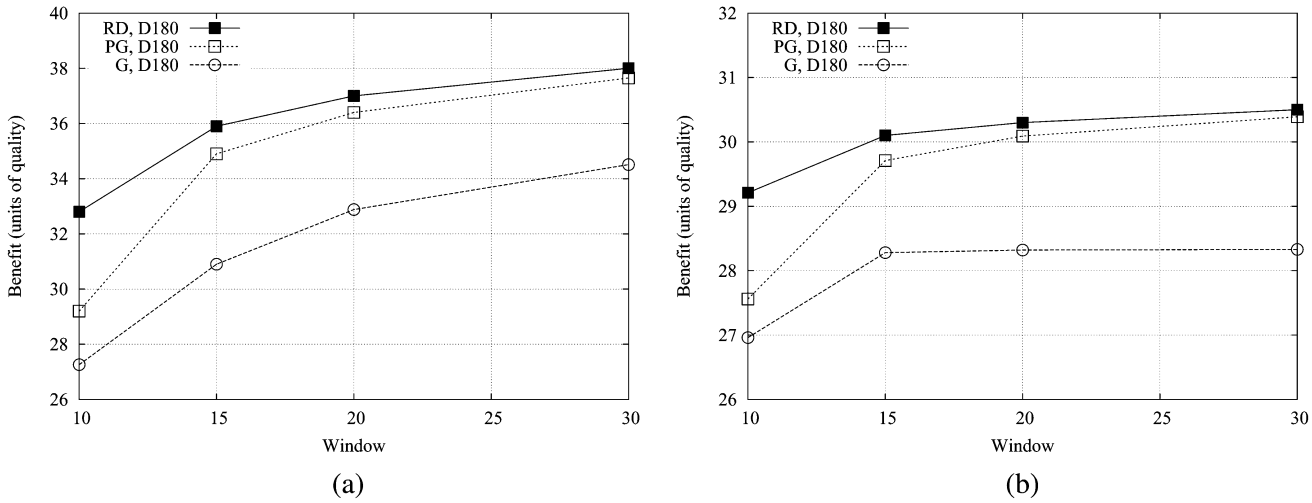

Fig. 3. Performance of the streaming algorithms (in units of quality) versus maximal pre-fetch window size W_τ, with $\mu_F = \mu_B = 180$ ms, $\varepsilon_F = 0.2$, and $\varepsilon_B = 0$ and $\mathrm{Rate} = 6500$ bits/s. (a) R11. (b) R21.

that is close to the one for the RD scheme. Extended simulations over a large range of channel parameters have confirmed this observation.

Fig. 3 analyzes the impact of the pre-fetching window size W_τ on the received quality at a constant channel bandwidth of 6500 bits/s, for the R11 and R21 distortion templates. The transmission delays are still defined by $\mu_F = \mu_B = 180$ ms and the maximal pre-fetch delay is defined in terms of W_τ by $\tau = W_\tau \times 50$ ms. We observe that, for small values of W_τ, the quality significantly increases with W_τ. When W_τ becomes sufficiently large, the end-to-end quality tends to saturate and does not improve significantly 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 a close to optimal RD performance. This sufficient pre-fetch delay is related to the time needed to receive feedback from the client.

Typically, PG needs a window W_τ larger than 15 to 20 frames, which corresponds to a maximum pre-fetch delay of about 750 to 1000 ms, as seen from Fig. 3. The average RTT considered in these simulations is 2×180 ms $= 360$ ms. We conclude that PG needs a pre-fetch delay of about two or three RTTs before being able to compete with ARC-RaDiO. In addition, note that ARC-RaDiO also needs two to three RTTs of pre-fetch delay before its performance saturates in regards to quality.

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

To evaluate the rate savings of PG relative to G, we plot in Fig. 5 the ratio between the average transmission rates required by G and PG in order 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/s, which corresponds to a medium-to-high quality, as

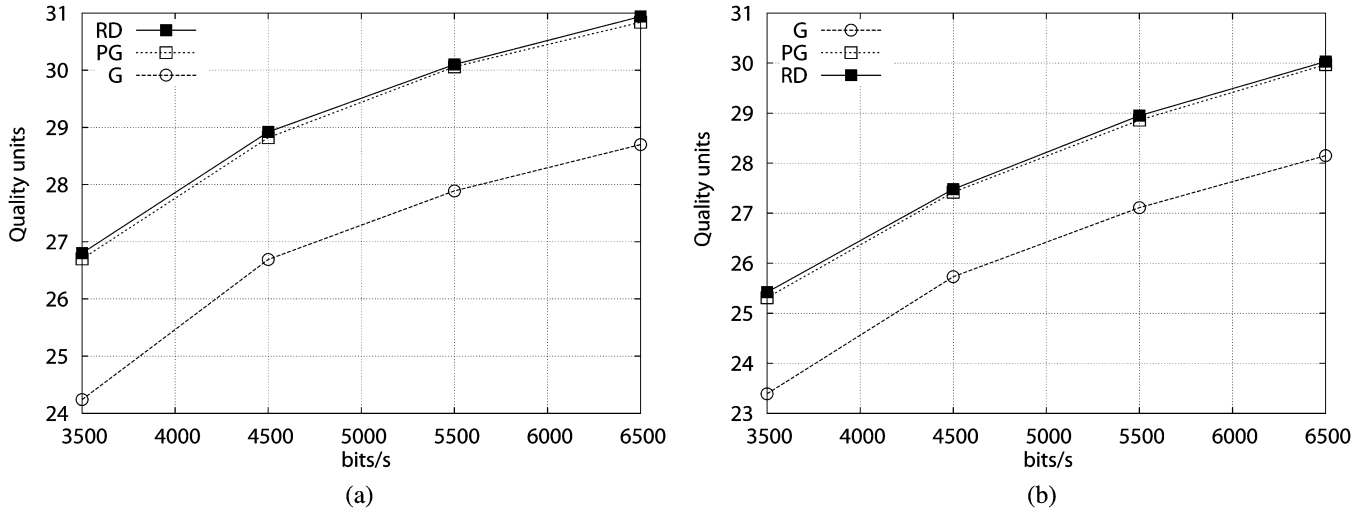

Fig. 4. Performance of the streaming algorithm (in units of quality), versus channel bit-rate (in bits/s), with $\mu_F = \mu_B = 60$ ms, $\varepsilon_F = 0.2$, $R21$ distortion template, and $W_\tau = 20$. (a) $\varepsilon_B = 0$. (b) $\varepsilon_B = 0.2$.

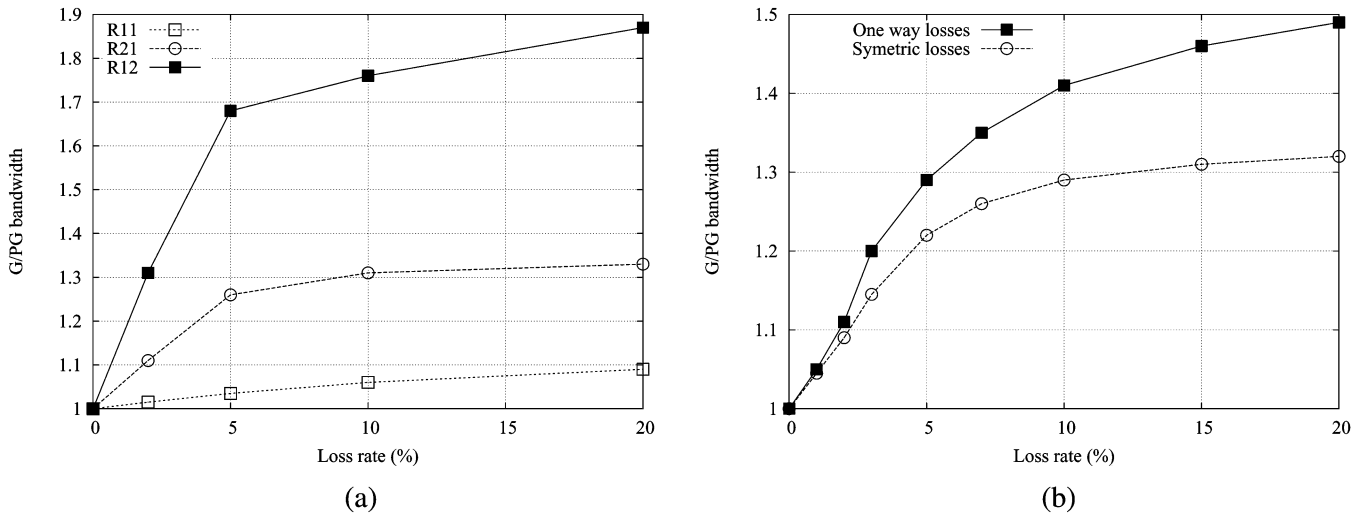

Fig. 5. Ratio between the G and PG transmission rates needed to achieve a constant quality, as a function of the loss probability ε_F, with $W_\tau = 20$. (a) $\varepsilon_B = 0$, $\mu_{F,B} = 180$ ms. (b) R21, $\mu_{F,B} = 60$ ms.

the total encoding rate is equal to 5 kbits/s (20 fps $\times$ 5 layers $\times$ 50 bits/layer). Fig. 5(a) considers a lossless backward channel, and compares the rate ratio of G and PG for different distortion templates for the media data layers. 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 $R11$ result in a relatively good behavior of the greedy algorithm, while others like $R12$ or $R21$ cause a lot of penalizing unnecessary retransmissions. Fig. 5(b) analyzes the impact of the 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.

Fig. 6 analyzes the impact the loss rate ε_F and of the average RTT $\mu_F + \mu_B$ on the ratio between the rates consumed by G and PG to achieve the same average quality. The targeted quality is the one obtained by PG at 4.5 kbits/s. Not surprisingly, we observe that the benefit of PG over G increases with ε_F. However, we note that the gain of PG over G remains significant at small loss rates. Up to 20% of bit rate is saved when $\varepsilon_F = 3\%$. Besides, we observe that the benefit of PG over G sharply decreases once the RTT becomes smaller than 100 ms, that is, once the RTT gets smaller than the inter-frame interval. At that point, feedback information is received so rapidly that G does not have a chance to trigger a retransmission before the arrival of feedback. Hence, G and PG tend to behave identically as $\mu_F + \mu_B$ tends towards zero.

In Figs. 7 and 8, we analyze the consequences of a mismatch between the actual behavior of the transmission channel and its modeling by the scheduler in terms of the experienced packet delays and losses. Remember that the scheduler assumes independent and identically distributed (i.i.d.) losses, and that the average forward and backward trip times are, respectively, defined by μ_F and μ_B. We consecutively consider a non-i.i.d. loss model, and an inaccurate estimation of the average transmission delays in each direction.

In Fig. 7, the forward channel is characterized using a Gilbert loss model. The backward channel is assumed to be loss free. As before, the PG and G schedulers are based on the knowledge

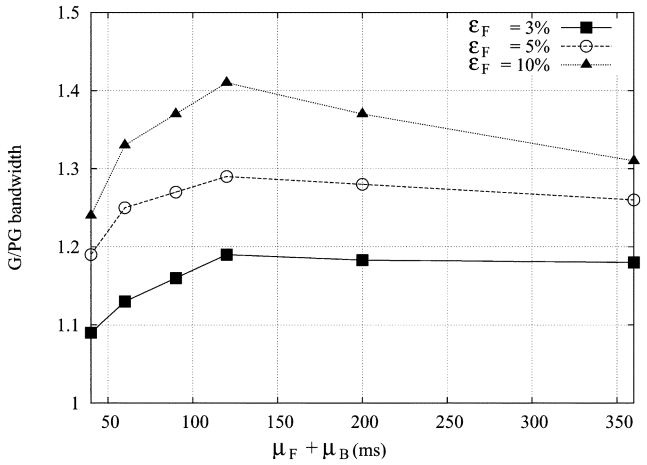

Fig. 6. Ratio between the G and PG transmission rates needed to achieve a constant quality, as a function of the RTT $\mu_F + \mu_B$, with $W_\tau = 20, \varepsilon_B = 0$ and R21 formatted content.

of the average loss rate ε_F, and do not exploit any additional knowledge about the loss generation model. The graphs in Fig. 7 plot the end-to-end performances of PG and G as a function of the average burst length of the Gilbert model for two different ε_F values. We observe that the increased burstiness equally penalizes G and PG. Hence, we conclude that the benefit of PG over G is maintained in the presence of more bursty packet loss processes. It is worth mentioning that *a priori* knowledge about the loss generation model could potentially be exploited to improve the streaming performance. However, such an investigation is beyond the scope of the present paper, and is left for future research.

In Fig. 8, we analyze the impact of inaccurate estimation of the forward and backward trip times. We denote $\hat{\mu}$ to be the estimate of $\mu_F = \mu_B$ used by the G and PG schemes in their scheduling algorithms. The graphs in Fig. 8 depict the streaming performance for both G and PG as a function of $\hat{\mu}$, for two different sizes of the prefetching window W_τ. Several observations are important to be noted. First, we observe that PG achieves optimal performance when $\hat{\mu} = \mu_{F,B}$, but that the performance of G slightly improves when $\hat{\mu}$ overestimates $\mu_{F,B}$. That is because G decides not to perform some of the retransmissions due to the RTT overestimation, and this partly compensates the loss in performance caused by premature retransmissions. Secondly, we observe that the performance of G decreases faster than the one of PG in case of round-trip time under-estimation, i.e., when $\hat{\mu} < \mu_{F,B}$. Therefore, it appears that by adopting more conservative scheduling rules, PG is less sensitive to parameter estimation inaccuracies relative to G.

Thirdly, by comparing Fig. 8(a) and (b), we note that in the case of RTT overestimation, the behavior of G and PG strongly depends on the size of the prefetching window W_τ. In particular, for a sufficiently large window, typically W_τ larger than two or three times the RTT estimate, we observe that the performances of G and PG decrease gradually as the RTT overestimation increases. In contrast, when W_τ becomes smaller than 2 RTT estimates, we observe that the performances of G and PG sharply decrease and coincide. For example, such a behavior is observed in Fig. 8(b) when $\hat{\mu} > 200$ ms, which corresponds to an RTT estimate larger than 400ms (i.e., larger than $\tau/2.5$ for the prefetch delay $\tau = 1000$ ms). In this case, due to the RTT overestimation, both schedulers wait exceedingly long before triggering a retransmission, so that neither G nor PG gets an opportunity to retransmit a packet more than once. Moreover, here the RD analysis does not encourage PG to postpone retransmissions as they approach the corresponding data unit deadlines. Hence, PG reduces to G once the prefetching window becomes small compared to the RTT estimate.

In summary, we conclude that a sufficiently large pre-fetching window is needed for PG to provide an improved streaming performance and increased robustness to inaccurate parameter estimation relative to G. Typically, the prefetch delay has to be larger than two or three RTT estimates in order to obtain the best performance out of PG.

C. Video Data

This section examines the RD performance for streaming actual packetized video content using the three scheduling algorithms under investigation. In the simulations, we employ the QCIF *Foreman* sequence encoded at 10 fps using JM2.1 of the JVT/H.264 compression standard [22]. In particular, 120 frames of the *Foreman* sequence are encoded with a constant quantization parameter for an average Y-PSNR of 35.86 dB and an average rate of 82.25 kbps. The 120-frame video segment is divided into six groups of pictures (GOPs), each GOP being composed of an I frame followed by 19 consecutive P frames. In the simulations we employ a concatenation of 20 video segments, each segment being composed of the 120 encoded 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 Y-PSNR values. We have observed that the PG, PG-AL, and ARC-RaDiO algorithms are more stable than G with respect to fluctuations of the communication channels, i.e., the PSNR values measured on distinct segments fluctuate less for PG and ARC-RaDiO than those for G.

The play-out delay of the video content at the receiver is set to 1 s. Furthermore, previous frame-error concealment on missing video frames is performed by the client application at the receiver. On the scheduler side, this concealment strategy is approximated as described in Appendix A and as proposed originally in [23], i.e., by assuming that a nondecodable frame is reconstructed based on the last decodable frame. Here, a frame is said to be decodable when the frame and all of its ancestors have been received on time. No concealment is considered between consecutive segments of 120 frames.

Fig. 9 considers a symmetric transmission channel in terms of packet loss and delay, and shows the average Y-PSNR values of the decoded video frames at the receiver as a function of the data rate on the forward channel for the G, the PG, the PG-AL, and the RD algorithms. Fig. 9(a)–(c) consider, respectively, prefetching window W_τ of 10, 20, and 30 frames.

In all the graphs, we observe that the conventional G algorithm performs significantly worse than the RD optimal or PG scheduling schemes. In addition, we also observe that PG-AL performs better than PG. Therefore, for actual video data, it is beneficial to exploit the *a priori* information inferred

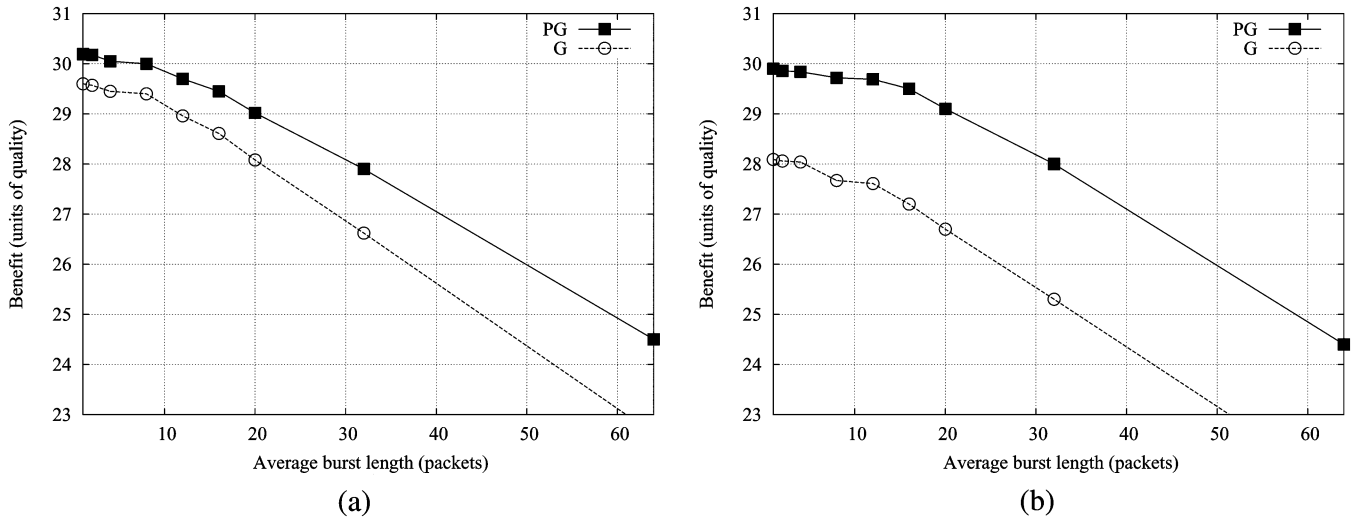

Fig. 7. Performance of the streaming algorithm (in units of quality), versus the average burst length of a Gilbert loss model. R21 content is considered, $\varepsilon_B = 0$, $\mu_{F,B} = 60$ ms, $W_\tau = 20$. (a) $\varepsilon_F = 0.05$. (b) $\varepsilon_F = 0.1$.

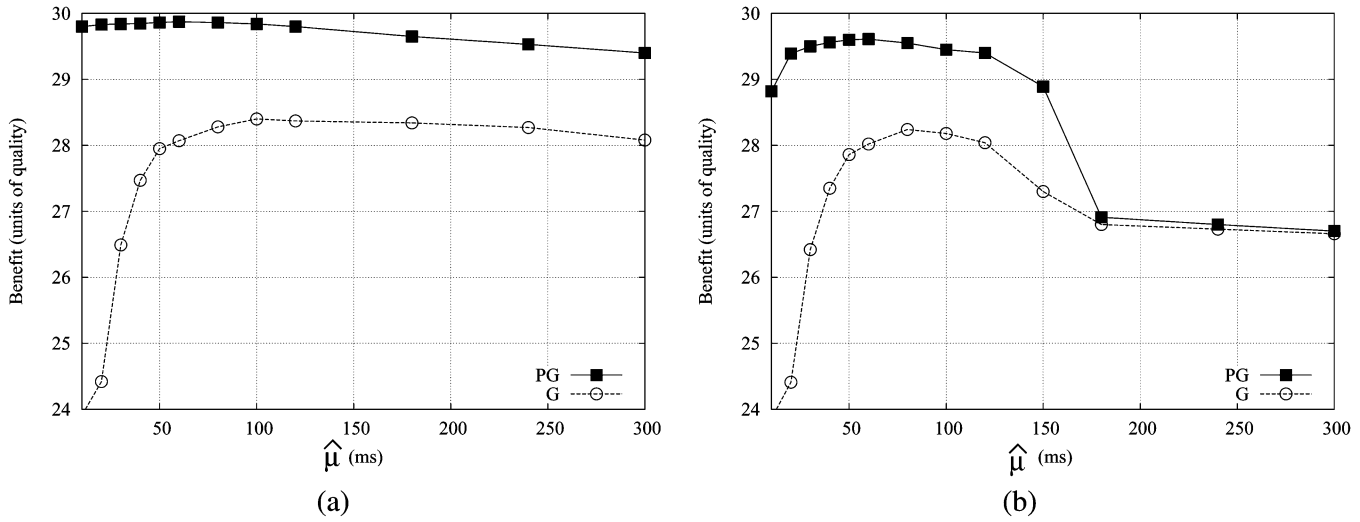

Fig. 8. Performance of the streaming algorithm (in units of quality), versus the FTT and BTT estimate $\hat{\mu}$. R21 content is considered, $\varepsilon_F = 0.1$, $\varepsilon_B = 0$, $\mu_{F,B} = 60$ ms. (a) $W_\tau = 20$. (b) $W_\tau = 10$.

about the probabilities of arrival based on transmissions of previous GOPs. This advantage can be explained as follows. The 20 frames GOP covers a time interval equal to 2 s, which is of the same order of magnitude as the authorized pre-fetch delay. As a consequence, only one or two GOPs at most are considered for transmission simultaneously. In addition, due to the dependencies within a GOP created at encoding, once a frame is expected to be lost, all its subsequent P frames are expected to be impossible to decode.

Hence, greedy mechanisms such as G and PG, which estimate the gain provided by a (re)transmission of a data unit based only on the transmission history of its dependent data units, block once the ACK of an ancestor data unit is lost or delayed. In that case, the scheduler can not afford to wait any longer 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 a premature retransmission impairs the overall RD performance if the corresponding 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 for transmissions of these descendant data units, which in turn permits to increase the waiting time in order to receive more knowledge about the arrival status of the ancestor data unit at the receiver, via acknowledgement packets. Here, it is worth pointing out 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 unit with a larger expected benefit from another GOP.

By comparing the plots in Fig. 9(a)–(c), we observe that for all scheduling algorithms the streaming performance improves as W_τ increases, but progressively saturates once W_τ becomes larger than the GOP size (i.e., $W_\tau > 20$). We explain such a

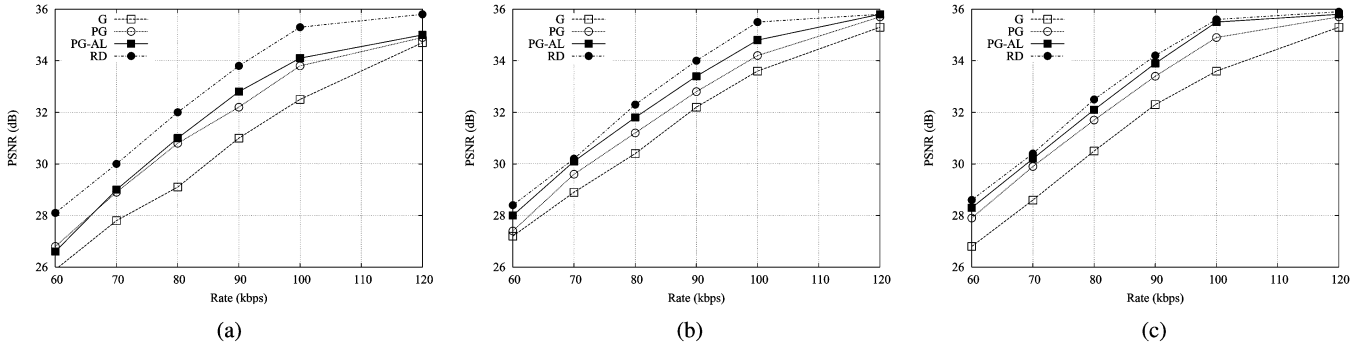

Fig. 9. PSNR quality versus bit rate, when streaming the QCIF Foreman video sequence. The channel conditions are defined by $\mu_F = \mu_B = 100$ ms, $\varepsilon_F = \varepsilon_B = 0.1$. Channel bandwidth is constant along the time. (a) $W_\tau = 10$. (b) $W_\tau = 20$. (c) $W_\tau = 30$.

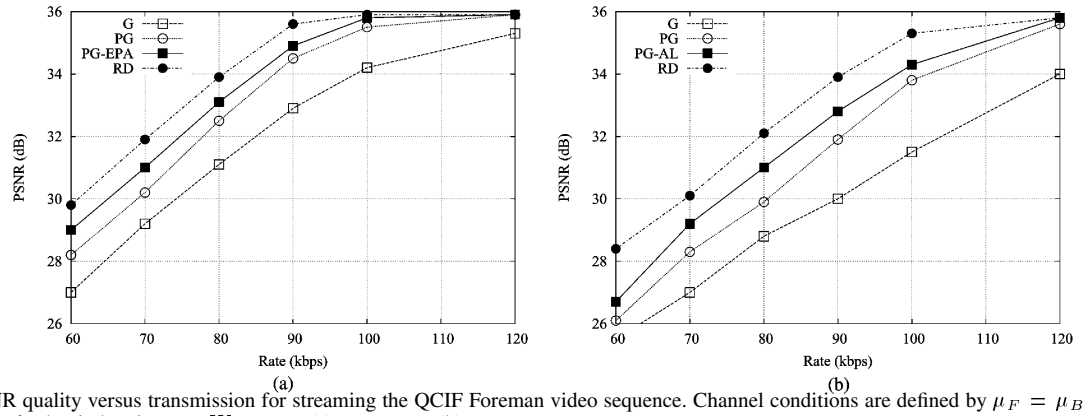

Fig. 10. Y-PSNR quality versus transmission for streaming the QCIF Foreman video sequence. Channel conditions are defined by $\mu_F = \mu_B = 100$ ms, and $\varepsilon_B = 0$. The pre-fetch window is set to $W_\tau = 20$. (a) $\varepsilon_F = 0.1$. (b) $\varepsilon_F = 0.2$.

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 Fig. 9(b) and (c) also reveals that PG algorithms approach 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 is nearly equal to that of ARC-RaDiO for $W_\tau = 30$. We conclude that PG algorithms are able to achieve close to RD optimal performance as long as the pre-fetch delay: 1) is larger than two or three RTTs (see Section V-B) and 2) is large enough to simultaneously schedule more than one group of interdependent data units.

For completeness, Fig. 10 considers a channel with reliable feedback (i.e., there is no loss on the reverse path), and confirms the above conclusions. Again, we observe that PG-AL provides improvement over PG and achieves performance that is much closer to the RD bound than the one obtained for the conventional G algorithm.

Fig. 11 shows the gain in Y-PSNR provided by PG-AL over G as a function of the loss rate ε_F and for different round-trip time (RTT) values. The video content is streamed at two different rates of 70 and 90 kbps, respectively. We first observe that the benefit of PG-AL over G decreases when the loss rate decreases. Nonetheless, we note that the gain still remains significant (around 1 dB) even for error rates that are as small as 3%. Regarding the impact of the channel average round-trip time $\mu_F + \mu_B$, we observe that the benefit of PG-AL over G increases with the RTT.

In order to understand better the influence of the RTT, it should be pointed out that the ratio between the RTT and the time that is required to forward a packet across the channel, is roughly proportional to the number of data units that can be transmitted within one RTT period, on the average. At the same time, note that the average RTT corresponds roughly to the period of time that is needed to draw a reliable conclusion about the arrival status of a data unit transmission, for the experimental setup considered here. Hence, based on these observations we conclude that the gains of PG-AL over G depend more on the average ratio between the RTT and the time required to forward a packet across the channel (excluding the queueing delay at the receiver due to cross-traffic), rather than on the absolute value of the average RTT. That is because the main distinction between PG-AL and G lies in the fact that PG-AL discourages retransmissions during the RTT feedback uncertainty period, while G on the other hand may trigger one such retransmission at one of the transmission opportunities (if there are any) during that RTT period.

In particular, let ΔT denote the average time interval between retransmissions. We approximate ΔT with the inter-frame period of the video encoding, which for a frame rate of 10 fps is equal to 100 ms. The approximation allows us then to estimate the ratio between the average RTT and ΔT for the experimental results shown in Fig. 11. Specifically, this ratio is one, two, and three, respectively for the average RTT values of 100, 200, and 300 ms. Keeping these values in mind, we can now remark regarding the results shown in Fig. 11 that the gains of PG-AL over G remain significant as long as the average RTT is longer

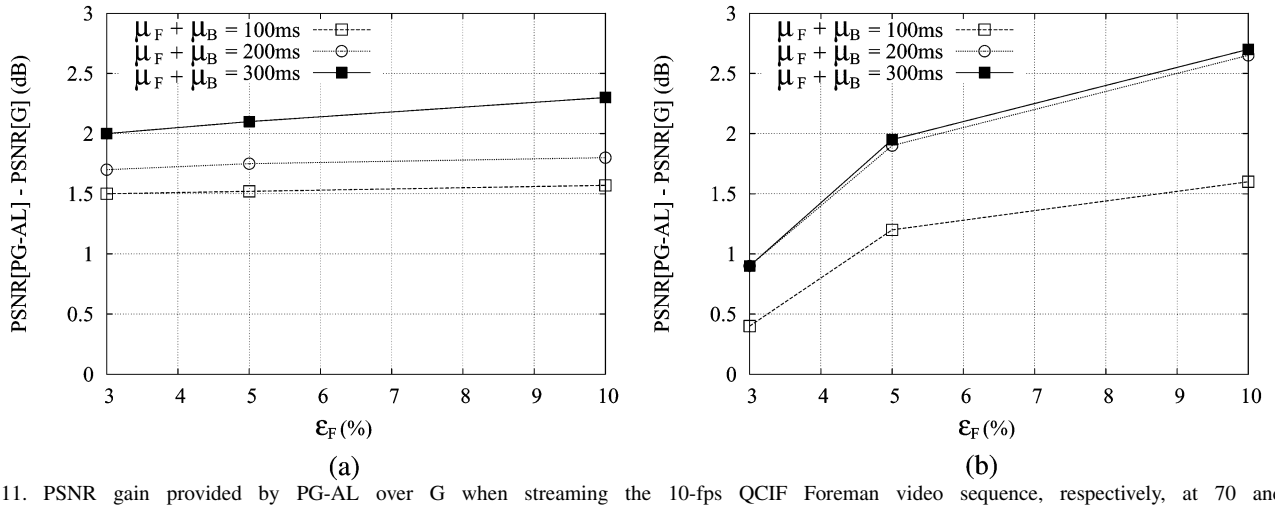

Fig. 11. PSNR gain provided by PG-AL over G when streaming the 10-fps QCIF Foreman video sequence, respectively, at 70 and 90 kbps. The gain is expressed in decibels (dB), and is plotted as a function of the loss rate ε_F, for different values of the average RTT $\mu_F + \mu_B$. The streaming and channel conditions are defined by $W_\tau = 20$, $\varepsilon_B = 0$. (a)Rate $= 70$ kbps. (b) Rate $= 90$ kbps.

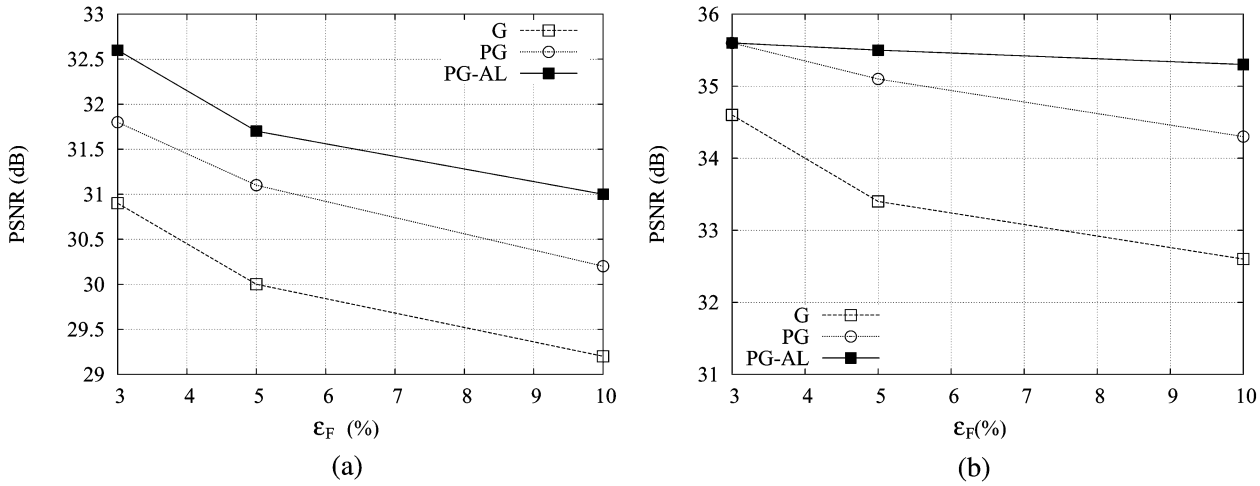

Fig. 12. PSNR quality versus loss rate ε_F, when streaming the QCIF Foreman video sequence at 70 and 90 kbps. The channel conditions are defined by $\mu_{F,B} = 100$ ms, and $\varepsilon_B = 0$. The pre-fetch window is set to $W_\tau = 20$. (a)Rate $= 70$ kbps. (b) Rate $= 90$ kbps.

than the time needed to forward a packet across the channel. On the other hand, a considerable improvement in performance of PG-AL over G should not be expected when the time for receiving feedback becomes shorter than the time needed to forward a data unit, on the average. In this latter case, the channel feedback can be considered as being instantaneous, i.e., at each novel transmission opportunity the potential ACKs related to all previous transmissions (except the very last one) have been received.

Finally, it should be mentioned that the performance of PG-AL is less sensitive to packet losses on the forward channel when streaming at 90 kbps relative to the 70 kbps case (see Fig. 12). In particular, for sufficiently large channel data rates PG-AL is able to overcome the effects of packet loss by employing properly selected retransmissions, thereby achieving a close to optimal video quality over a wide range of packet loss rates. Therefore, the results in Fig. 11(b) correspond approximately to the loss in performance of G relative to the optimal case of 35.86 dB, as a function of the packet loss rate on the forward channel.

In summary, the simulation results presented in this section lead to the following main conclusions.

- The proposed PG algorithms significantly outperform conventional greedy schemes.
- For sufficiently large values of the pre-fetch delay, the PG algorithms provide performances that are similar to the one of the RD optimal, but computationally complex, ARC-RaDiO scheme. Hence, the PG mechanism appears to be able to capture the essence of RaDiO algorithms, while preserving the simplicity of greedy approaches.
- The PG algorithm improves by estimating the *a priori* probabilities of arrival for data units. We have observed in our simulations that the benefit of PG-AL over PG is significant only when no more than two GODs can be 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 of its descendants are useless until the arrival status of this data is confirmed, and because, besides these descendant data units, there are no other data units to consider for transmission. In contrast, when the pre-fetching window contains a large number of GODs, it is always possible for a scheduler to switch between GODs so that it can wait sufficiently long to receive confirmation about the reception

of ancestor data units, prior to triggering their potentially unnecessary retransmission. This case was encountered in Section V-B when streaming artificial pre-formatted video data.

VI. Conclusions

This paper presents a low-complexity and rate-adaptive scheduling algorithm, which provides a close-to-optimal RD performance for streaming packetized media data. We study the suboptimality 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 PG scheduling algorithm to alleviate the distortion penalty due to greedy scheduling, while preserving the low computational complexity of these approaches. PG algorithms provide a RD performance that is close to optimal as long as the pre-fetch delay: 1) is larger than a few average round-trip times on the communication channel and 2) 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 PG algorithm however degenerates to a conventional greedy scheduling scheme. Simulation results show that for streaming packetized video content encoded at medium quality (35 dB) and on a channel characterized by 5% loss rate, the PG algorithm typically provides a 2-dB improvement in end-to-end RD performance relative to conventional greedy schedulers, with a comparable computational complexity. PG schemes therefore provide efficient and low-complexity solutions for practical streaming scenarios, in particular for on-demand media applications.

Appendix
Error Concealment

Sections II-B, III-A, and IV reveal that the decision to schedule a data unit directly depends on the amount by which the quality is expected to increase (or equivalently the distortion is expected to decrease) if the data unit reaches the receiver on time, given the past transmission decisions. In the following, we denote the sensitivity Q_l^t to be the increase in quality (or decrease in distortion) expected at time t from the correct reception of data unit l. We define Q_l^t both in the absence and in the presence of error concealment mechanisms.

In the absence of an error-concealment strategy, we define an incrementally additive distortion model for which 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. Hence, ΔD_l reflects the increment in quality obtained from data unit l, assuming all its ancestors have been correctly received. If one of the ancestors of l has not been received, no gain in quality is obtained from reception of data unit l. A priori, ΔD_l can be defined based on any distortion or quality metric. In our video streaming simulations, ΔD_l is defined as a Y-PSNR increase (in decibels). For the incrementally additive distortion model, the sensitivity Q_l^t is defined by [7]

$$Q_l^t = \sum_{l' \succeq l} \left(\Delta D_{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'') \right) \tag{22}$$

where $p_c^t(i)$ denotes the probability that data unit i reaches the client on time given the transmission decisions performed for data unit i up to time t. In particular, in the greedy case $p_c^t(i) = p\left(i \mid \varkappa_i^t\right)$, while in the case of RaDiO we have $p_c^t(i) = 1 - \epsilon\left(\pi_i^t\right)$, where π_i^t denotes the list of transmission actions performed on data unit i up to (and including) time t.

In the presence of concealment, the incrementally additive model does not hold anymore. The decrease in distortion if data unit l is decoded depends on which other data units have been decoded [7]. The purpose of this Appendix A is to explain how the sensitivity Q_l^t is computed in the presence of error concealment. We follow the approach introduced by Chakareski and Girod in [23]. 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 selects 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 $j \in \Gamma_l$. Again, ΔD_l^j is defined in dBs, i.e., in terms of PSNR increase. 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

$$P^t[i \leftarrow j \mid j] = \prod_{k \in \chi(j,i)} \left(1 - p_c^t(k)\right). \tag{23}$$

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

$$\begin{aligned} Q_l^t = & \sum_{l' \succeq l} \sum_{i: l' \in \Gamma_i} \Delta D_i^{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'') \\ & \times \prod_{k \in \chi(l',i)} \left(1 - p_c^t(k)\right) \\ & - \sum_{i: l \in \Gamma_i} \sum_{j \in \Gamma_i : l \in \chi(j,i)} \Delta D_i^j \prod_{j' \preceq j} p_c^t(j') \\ & \times \prod_{k \in \chi(j,i), k \neq l} \left(1 - p_c^t(k)\right). \end{aligned} \tag{24}$$

The first term in (24) 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.

Equation (24) is generic, and holds for any kind of dependency among data units. We now develop (24) to make it explicit for the case of interest in Section V-C. There, the Foreman video sequence is encoded as a succession of GOPs.

Each GOP contains 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 (24) becomes

$$Q_l^t = \left(\sum_{l' \succeq l} \sum_{i=l'}^{l'+N_c} \Delta D_i^{l'} \prod_{l'' \preceq l', l'' \neq l} p_c^t(l'') \times \prod_{k \in \chi(l',i)} (1 - p_c^t(k)) \right) - C_l^t \quad (25)$$

where C_{l}^{t} is given by the equation shown at the bottom of the page. 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 such that $l < j \leq i$.

REFERENCES

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

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

[3] M. Zhourong and A. Ortega, "Fast adaptive media scheduling based on expected run-time distortion," in *Asilomar Conf. Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2002.

[4] 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 *Proc. IEEE WCNC*, Atlanta, GA, Mar. 2004.

[5] D. Li, C. Gene, C. Chen-Nee, and S. J. Y. Ben, "Joint server/peer receiver-driven rate-distortion optimized video streaming using asynchronous clocks," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Singapore, Oct. 2004.

[6] P. A. Chou and Z. Miao, Rate-Distortion Optimized Streaming of Packetized Media Microsoft Research, Redmond, WA, 2001, Tech. Rep. MSR-TR-2001-35.

[7] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, Apr. 2005.

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

[9] ——, "Branch and bound algorithms for rate-distortion optimized media streaming," *IEEE Trans. Multimedia*, vol. 8, no. 1, pp. 170–178, Feb. 2006.

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

[11] A. Sehgal, A. Jagmohan, O. Verscheure, and P. Frossard, "Fast distortion-buffer optimized streaming of multimedia," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Genoa, Italy, Sep. 2005.

[12] J. Chakareski and B. Girod, "Server diversity in rate-distortion optimized media streaming," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Barcelona, Spain, Sep. 2003.

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

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

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

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

[17] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *IEEE Trans. Image Process.*, vol. 2, no. 4, pp. 160–175, Apr. 1993.

[18] J. Chakareski, J. Apostolopoulos, and B. Girod, "Low-complexity rate-distortion optimized video streaming," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Singapore, Oct. 2004.

[19] A. Ortega, "Optimal bit allocation under multiple rate constraints," in *Proc. IEEE Data Compression Conf. (DCC)*, Snowbird, UT, Apr. 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 J. Select. Areas Commun.*, 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 Int. Conf. Image Processing (ICIP)*, Singapore, Oct. 2004.

[22] Video Coding for Low Bitrate Communication Telecom. Standardization Sector of ITU, 2003, Draft ITU-T Recommendation H.264.

[23] J. Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *IEEE Data Compression Conf. (DCC)*, Snowbird, UT, Mar. 2003.

Christophe De Vleeschouwer was born in Namur, Belgium, in 1972. He received the M.S. and Ph.D. degrees from the Université Catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, in 1995 and 1999, respectively.

He was a Research Engineer with the IMEC Multimedia Information Compression Systems Group (1999–2000). He has been a Visiting Research Fellow at the University of California at Berkeley (2001–2002) and a Postdoctoral Researcher at Ecole Polytechnique Fédérale de Lausanne (2004). He is now a Belgian National Science Foundation (NSF) Research Associate at UCL in the Communication and Remote Sensing Laboratory (TELE). His research interests include video and image processing for communication and networking applications, including content management and security issues, in addition to nonlinear signal expansion techniques and their use for signal analysis and signal interpretation.

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

Jacob Chakareski received the B.S. degree from Ss. Cyril and Methodius University, Skopje, Macedonia, in 1996, the M.S. degree from Worcester Polytechnic Institute, Worcester, MA, in 1999, and the Ph.D. degree from Rice University, Houston, TX, in 2005, all in electrical and computer engineering. He performed his doctoral thesis research at Stanford University, Palo Alto, CA from 2002 to 2005.

In 2005-2006, he was a Postdoctoral Researcher with the Swiss Federal Institute of Technology (EPFL), Lausanne. Presently, he is a Senior Engineer with Layered Media, Inc., Rochelle Park, NJ, where he works on real-time video communication. He has held research positions with Microsoft and Hewlett-Packard. He has co-authored over 50 international publications and has two pending patent applications. His field of interests are networked media systems, distributed computation and control, wireless communications, and Macedonian history.

Dr. Chakareski received the best student paper award at the IS&T/SPIE VCIP 2004 conference.

Pascal Frossard (S'96–M'01–SM'04) received the M.S. and Ph.D. degrees, both in electrical engineering, from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1997 and 2000, respectively.

Between 2001 and 2003, he was a Member of Research Staff at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he worked on media compression and streaming technologies. Since 2003, he has been an Assistant Professor at EPFL, supported by the Swiss National Science Foundation. His research interests include image representation and coding, nonlinear representations, visual information analysis, joint source and channel coding, multimedia communications, and multimedia content distribution.

Dr. Frossard is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2004-present) and of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006-present). Since 2004, he has served as Vice Chair of the IEEE Multimedia Communications Technical Committee, and as a member of the IEEE Multimedia Signal Processing Technical Committee. He received the Swiss National Science Foundation Professorship Award in 2003, and the IBM Faculty Award in 2005.

Dr. Frossard has been the General Chair of IEEE ICME 2002 (Lausanne, Switzerland) and a member of the organizing or technical program committees of numerous conferences. He has served as Guest Editor of Special Issues on Streaming Media (IEEE TRANSACTIONS ON MULTIMEDIA), on Media and Communication Applications on General Purpose Processors: Hardware and Software Issues (*Journal of VLSI SPSS*), and on Image and Video Coding Beyond Standards (*Journal of Signal Processing*). He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2004-present) and of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006-present). He served as a member of the Editorial Board of the *EURASIP Journal of Signal Processing* (2003–2005). Since 2004, he has served as Vice Chair of the IEEE Multimedia Communications Technical Committee, as a member of the IEEE Multimedia Signal Processing Technical Committee, and of the IEEE Multimedia Systems and Applications Technical Committee. He received the Swiss National Science Foundation Professorship Award in 2003, and the IBM Faculty Award in 2005.