SCHOOL OF ENGINEERING - STI
SIGNAL PROCESSING INSTITUTE
Jacob Chakareski and Pascal Frossard

CH-1015 LAUSANNE
Telephone: *+4121 6936874*
Telefax: *+4121 6937600*
e-mail: `jakov.cakareski@epfl.ch`

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

DISTRIBUTED COLLABORATION FOR ENHANCED SENDER-DRIVEN VIDEO STREAMING

Jacob Chakareski and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Signal Processing Institute - Technical Report

TR-ITS-2007.008

May 8th, 2007

Part of this work has been submitted to IEEE TMM

This work has been supported by the Swiss National Science Foundation, under grant PP002-68737.

Distributed Collaboration for Enhanced Sender-Driven Video Streaming

Jacob Chakareski and Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute
1015 Lausanne, Switzerland
{jakov.cakareski,pascal.frossard}@epfl.ch

Abstract

We propose a sender-driven system for adaptive streaming from multiple servers to a single receiver over separate network paths. The servers employ information in receiver feedbacks to estimate the available bandwidth on the paths and then compute appropriate transmission schedules for streaming media packets to the receiver based on the bandwidth estimates. An optimization framework is proposed that enables the senders to compute their transmission schedules in a distributed way, and yet to dynamically coordinate them over time such that the resulting video quality at the receiver is maximized. To reduce the computational complexity of the optimization framework an alternative technique based on packet classification is proposed. The substantial reduction in online complexity due to the resulting packet partitioning makes the technique suitable for practical implementations of adaptive and efficient distributed streaming systems. Simulations with internet network traces demonstrate that the proposed solution adapts effectively to bandwidth variations and packet loss. They show that the proposed streaming framework provides superior performance over a conventional distortion-agnostic scheme that performs proportional packet scheduling on the network paths according to their respective bandwidth values.

I. Introduction

Collaborative streaming has drawn considerable attention in recent years. One of the scenarios that fall into this category is distributed streaming, where multiple senders transmit packets over separate network paths to a single receiver. This setting can be encountered for example in Content Delivery Networks (CDNs), where multiple streaming or edge servers may stream multimedia data to a single client, or in peer-to-peer (P2P) overlay networks, where a client may have access to the same multimedia data at multiple peers in the network. A related concept is the Digital Fountain model [1] where the system tries to minimize the download time of a file at a client by connecting to multiple mirror server sites.

The possibility to receive the same data over multiple paths increases the resilience of the media presentation to network outages or congestion onsets. These may occur on some of the paths and thereby may prevent the timely delivery of the data units sent exclusively over those individual paths. Furthermore, if the network paths exhibit good transmission quality it may be still desirable to spread the transmissions of the media data over multiple paths, i.e., to send different data units over different paths, in order to reduce the start-up delay of the client application and to ensure smooth and continuous play-out. The adaptive streaming session can be controlled either by the receivers, or by the senders. The latter solution provides several advantages in terms of media optimization, since the relative importance of the media packets can be known at the servers. In addition, the deployment of practical solutions is facilitated in this case since the management of overall network resources becomes easier, and the clients do not require any complex or non-standard functionality to access the streaming application.

We propose in this paper a distributed sender-driven streaming solution where servers collaboratively adapt to the network status in order to provide the media client with a superior video quality. Instead of computing transmission schedules for every sender as in receiver-driven approaches, the client only monitors incoming packets and sends back information about the network availability. This information is distributed to all the servers, such that they can coordinate the delivery of the media stream, and avoid wasting bandwidth resources. Bandwidth estimates are then used in conjunction with a rate-distortion optimization framework to compute appropriate transmission actions at each sender. In essence, this framework enables the senders, based on the feedback information from the client, to compute independently, yet in a coordinated fashion, what their respective transmission policies should be, given the available bandwidth on each network path. In order to maximize the resulting video quality at the receiver, each sender takes into account the relative distortion importance of every video packet when computing its transmission policy.

The computational load imposed by the global optimization framework, however, increases with the number of servers, which may be prohibitive in practical scenarios. Therefore, we propose an alternative algorithm based on a priori packet classification. The technique achieves similar performance with a dramatically reduced complexity, thereby providing an interesting solution for practical and scalable implementations of adaptive and efficient distributed streaming systems. Simulation results on real internet

This work has been supported in part by the Swiss National Science Foundation under grant PP002-68737.

traces demonstrate the superiority of the proposed rate-distortion optimized system compared to distortion-agnostic streaming solutions.

To the best of our knowledge, the earliest work that studied the problem of transmission coordination among the multiple senders in distributed streaming is [2]. In this work, the authors propose an algorithm that is run at the client and that performs rate allocation and packet partitioning among the senders. Based on this information the servers then adjust their sending rates in order to meet constraints from the network, whose aggregated bandwidth still allows to transmit the complete media stream. In a follow-up work, the authors combined the previously proposed algorithm with forward error correction for improved error resilience to burst packet loss [3]. Similarly, the works in [4, 5] consider receiver-driven control protocols that synchronize the senders' transmissions in a rate-distortion optimized way. For improved error-resilience, Multiple Description Coding (MDC) is employed at each sender to pre-encode (prior to transmission) a progressively encoded media content that is streamed afterwards to the client. Another related work is [6], where a rate-distortion optimization framework is proposed for packet scheduling in receiver-driven distributed video streaming. The paper establishes that the gains in performance due to server (path) diversity, relative to a single server (path) case, are dependent on the quality of the network paths in terms of packet loss and delay. Finally, the works in [7, 8] examine the performance of an MDC scheme for distributed video streaming in CDNs. The authors report a 20-40% reduction in client video distortion, for the considered network conditions and topologies, relative to conventional CDNs that do not employ multiple description encoded video streams.

Differently from the prior work, we propose in this paper an optimization framework for sender-driven streaming, where multiple servers synchronize their transmission schedules for sending parts of a standard (single description) video stream under bandwidth and delay constraints. Each server selects the media packets that have to be transmitted in priority and determines the corresponding scheduling strategy in order to minimize the distortion at the receiver. The assistance from the receiver is limited to feedback necessary for bandwidth estimation, which is certainly required for the design of efficient adaptive streaming strategies.

The rest of the paper is organized as follows. In Section II, we describe the distributed streaming system, and formulate the rate-distortion optimization problem when servers collaborate to minimize the distortion at the client. Section III describes the media and network models, and the methodology that is used for computing the rate-distortion optimal packet scheduling strategy. A low-complexity solution is later proposed in Section IV, where a priori source pruning and packet classification allow to dramatically reduce the computational complexity relative to the original optimization framework. Finally, in Section V, simulation experiments on synthetic and realistic network traces show that both algorithms outperform a conventional distortion-agnostic system, which performs packet scheduling based only on the available bandwidth.

II. SENDER-DRIVEN DISTRIBUTED STREAMING

A. Media Communication Model

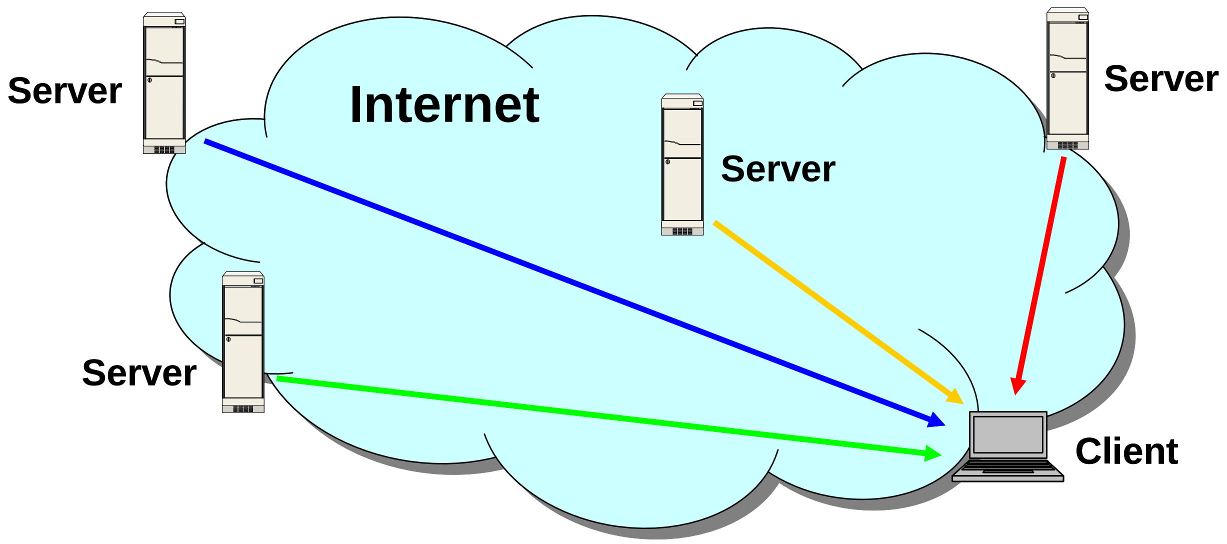

Fig. 1. Distributed streaming: multiple senders - single receiver.

We consider a streaming system with M senders (servers) transmitting a media presentation on M independent bidirectional paths to a streaming media client, as illustrated in Fig. 1. The media presentation is available at each server in the form of a collection of interdependent data units with delivery deadlines. The servers encapsulate media data units into packets and collaboratively stream (parts of) the media presentation to the streaming client. The client, in turn, monitors incoming packets, and piggybacks information about the availability of the network paths on acknowledgement packets to all the sending servers.

Based on this information, the servers periodically estimate the bandwidth available on all the paths from the senders to the media client. At the end of each estimation period ΔT the servers therefore know bandwidth estimates $\widetilde{R}_1, \ldots, \widetilde{R}_M$ for every path. This information is used to adapt the rate-distortion optimized streaming strategy to a varying network availability.

Based on bandwidth estimates, the servers then select the media packets to be transmitted along with the proper schedule. A transmission schedule or policy basically represents the actions performed by the servers on a given data unit l, at each transmission opportunity. More formally, let $t_0, t_1, \ldots, t_{N-1}$ be a window of N transmission opportunities at which the senders can transmit packets with the data unit l to the streaming client, prior to its delivery deadline (i.e., $t_i < t_{DTS,l}, \forall 0 \leq i \leq N-1$). The transmission policy π_l for data unit l then forms a matrix of binary actions a_{mi}, for $i = 0, \ldots, N-1$, and $m = 1, \ldots, M$, where the row and column indices denote respectively the corresponding sender/path and transmission opportunity at which that action is undertaken. In other words, $a_{mi} = 1$ signifies the transmission of a packet with the data unit l by the server m at transmission opportunity t_i, and $a_{mi} = 0$ signifies the converse. The arrival of a packet with the data unit at the client is immediately acknowledged to every sender by sending acknowledgement packets in the backward direction on all M paths. Note that sender m will only send a packet with the data unit l for $a_{mi} = 1$ if no acknowledgement arrives at the sender by t_i to report the correct reception of the data unit due to earlier transmissions.

B. *Optimization Problem*

We are interesting in finding the transmission policy that minimizes the distortion experienced at the client, under constraints given by the available bandwidth on each network path. Suppose that there are L data units in the multimedia session. Let π_l be the transmission policy for data unit $l \in \{1, \ldots, L\}$ and let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_L)$ be the vector of transmission policies for all L data units. Any given policy vector $\boldsymbol{\pi}$ induces for the multimedia session an expected distortion $D(\boldsymbol{\pi})$ and a vector of expected transmission rates $R(\boldsymbol{\pi}) = [R_0(\boldsymbol{\pi})\, R_1(\boldsymbol{\pi}) \ldots R_M(\boldsymbol{\pi})]$ on the forward channels of the network paths. We thus seek the policy vector $\boldsymbol{\pi}$ that minimizes the expected distortion $D(\boldsymbol{\pi})$ such that the expected transmission rate in the forward direction on every network path does not exceed the available bandwidth, i.e.,

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} D(\boldsymbol{\pi}), \qquad (1)$$
$$\text{s.t. } R_m(\boldsymbol{\pi}) \leq \widetilde{R}_m\,,\ m = 1, \ldots, M.$$

Using the method of Lagrange multipliers, we reformulate (1) as an unconstrained optimization problem. That is, we seek the policy vector $\boldsymbol{\pi}$ that minimizes the expected Lagrangian $J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \sum_{m=1}^{M} \lambda_m R_m(\boldsymbol{\pi})$ for some vector of positive Lagrange multipliers $\boldsymbol{\lambda} = [\lambda_1\, \lambda_2 \ldots \lambda_M]$, and thus achieves a point on the lower convex hull of the set of all achievable distortion-rate pairs $(D(\boldsymbol{\pi}), R(\boldsymbol{\pi}))$[1]. We solve the rate-distortion optimized distributed streaming problem in the next section, and we later propose a low complexity solution based on packet classification.

III. Rate-Distortion Optimized Distributed Streaming

A. *Media and Network Models*

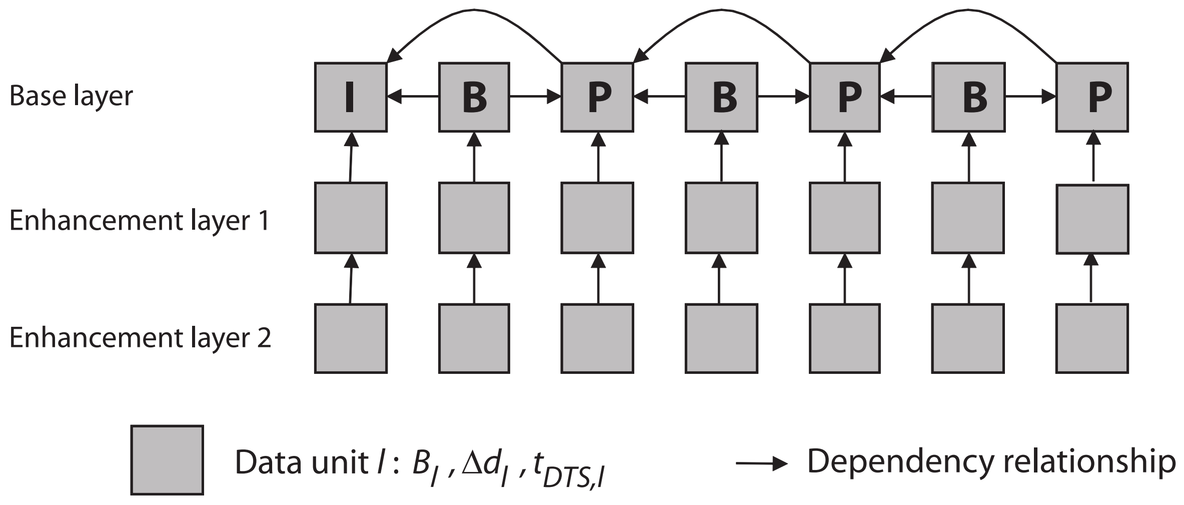

Fig. 2. Directed Acyclic Graph: Illustration of data unit interdependencies in a typical scalable media stream (I, P and B resp. represent Intra, Predicted and Bidirectional predicted frames in MPEG terminology).

[1]Equivalently, the set of all achievable distortion-rate $(D, R_1, \ldots, R_M)$ (M+1)-tuples.

We first overview the media and network models that we use to solve the rate-distortion optimized distributed streaming problem. These models are commonly accepted and have been widely used recently for the design of optimized streaming solutions. We limit ourselves to the minimal presentation necessary to understand the algorithms proposed in this paper, and refer the reader to [9, 10] for more details.

In a streaming media system, the encoded data are packetized into *data units* and stored in a file on a media server. We consider here non-scalable as well as layered streams proposed by recent standardization bodies, as opposed to multiple description video coding that is generally limited to two descriptions, hence two streaming servers. All the data units in the presentation have interdependencies, which can be expressed by a directed acyclic graph, as illustrated in Fig. 2. Each node of the graph corresponds to a data unit, where an edge of the graph directed from data unit l' to data unit l implies that data unit l' can be decoded only if data unit l is first decoded.

Associated with each data unit l is a size B_l, a decoding time $t_{DTS,l}$, a set of data units $\mathcal{N}_c^{(l)}$ and an importance $\Delta d_l^{(l_1)}$. Specifically, the size B_l is the size of the data unit in bytes. $t_{DTS,l}$ is the *delivery deadline* by which data unit l must arrive at the client to be usefully decoded. Packets containing data units that arrive after the data units' delivery deadlines are discarded. Furthermore, $\mathcal{N}_c^{(l)} = \{1, \ldots, l\}$ is the set of data units that the receiver considers for error concealment in case data unit l is not decodable by the receiver on time. Finally, $\Delta d_l^{(l_1)}$, for $l_1 \in \mathcal{N}_c^{(l)}$, is the reduction in reconstruction error (distortion) for the media presentation, when data unit l is not decodable but is concealed with data unit l_1 that is received and decoded on time.

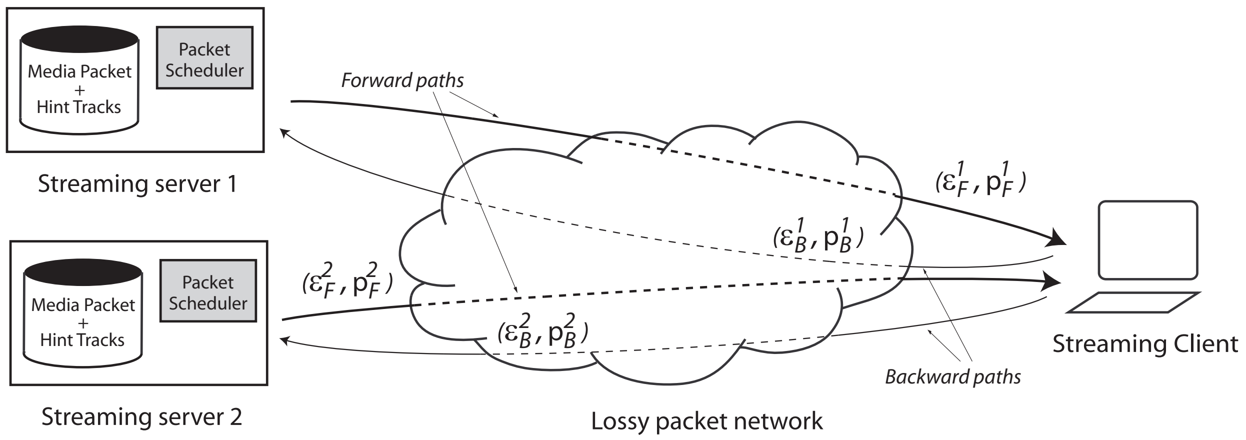
Media Packet + Hint Tracks
Packet Scheduler
Streaming server 1
Forward paths
(ε_F^1, p_F^1)
(ε_B^1, p_B^1)
Media Packet + Hint Tracks
Packet Scheduler
Streaming server 2
(ε_F^2, p_F^2)
(ε_B^2, p_B^2)
Streaming Client
Backward paths
Lossy packet network

Fig. 3. Framework for distributed streaming with packet erasure channels.

The network is represented as a simple model where the forward and backward directions on the network path between a sender (server) and the receiver are described as independent time-invariant packet erasure channels with random delay, as represented in Fig. 3. In general, each of the paths has different characteristics in terms of loss probability or latency. Hence, the path m (for $m = 1, \ldots, M$) is specified with the probabilities of packet loss ϵ_F^m and ϵ_B^m, and the probability densities of the transmission delay p_F^m and p_B^m, respectively. This means that if a server sends a packet on the forward channel m at time t, then the packet is lost with probability ϵ_F^m. However, if the packet is not lost, then it arrives at the client at time t', where the forward trip time $FTT^m = t' - t$ is randomly drawn according to the probability density p_F^m. Therefore, we let $P\{FTT^m > \tau\} = \epsilon_F^m + (1-\epsilon_F^m)\int_\tau^\infty p_F^m(t)dt$ denote the probability that a packet transmitted by the server at time t does not arrive at the client application by time $t+\tau$, whether it is lost in the network or simply delayed by more than τ. Similar relations can be defined for the backward channel, which induce a probability $\epsilon_R^m = 1-(1-\epsilon_F^m)(1-\epsilon_B^m)$ of losing a packet either on the forward or backward channel, and a round trip time distribution $P\{RTT^m > \tau\} = \epsilon_R^m + (1-\epsilon_R^m)\int_\tau^\infty p_R^m(t)dt$, where $p_R^m = p_F^m * p_B^m$ is the convolution of p_F^m and p_B^m. Note that $P\{RTT^m > \tau\}$ is the probability that the server m does not receive an acknowledgement packet by time $t+\tau$ for a data packet that has been transmitted at time t.

B. Expected distortion and streaming rate

Now that media and network models have been defined, we can compute the distortion $D(\boldsymbol{\pi})$ and the streaming rate $R(\boldsymbol{\pi})$ that results from streaming policy $\boldsymbol{\pi}$, by extending solutions proposed in [9, 10] to the multiple sender case. The expected transmission rate $R_m(\boldsymbol{\pi})$ on path m is the sum of the expected transmission rates on this path for each data unit $l \in \{1, \ldots, L\}$ in the presentation:

$$R_m(\boldsymbol{\pi}) = \sum_l B_l \rho_m(\pi_l), \tag{2}$$

where B_l is the size of data unit l in bytes and $\rho_m(\pi_l)$ is the *expected cost* per byte, or the expected number of transmitted bytes per source byte under policy π_l on path m. The expected distortion $D(\boldsymbol{\pi})$ is somewhat more complicated to express, but it can be expressed in terms of the *expected error*, or the probability $\epsilon(\pi_l)$ that data unit l does not arrive at the client on time (under policy π_l). It reads

$$\begin{aligned} D(\boldsymbol{\pi}) &= D_0 - \sum_l \sum_{l_1 \in \mathcal{N}_c^{(l)}} \Delta d_l^{(l_1)} \prod_{j \in \mathcal{A}(l_1)} (1-\epsilon(\pi_j)) \times \\ & \prod_{l_2 \in \mathcal{C}(l,l_1)} \left(1 - \prod_{l_3 \in \mathcal{A}(l_2) \backslash \mathcal{A}(l_1)} (1-\epsilon(\pi_{l_3})) \right) \end{aligned} \quad (3)$$

where D_0 is the expected reconstruction error for the presentation if no data units are received. $\mathcal{A}(l_1)$ is the set of ancestors of l_1, including l_1. $\mathcal{C}(l,l_1)$ is the set of data units $j \in \mathcal{N}_c^{(l)} : j > l_1$ that are not mutual descendants, i.e., for $j,k \in \mathcal{C}(l,l_1) : j \notin \mathcal{D}(k), k \notin \mathcal{D}(j)$, where $\mathcal{D}(j)$ is the set of descendants of data unit j, and "\" denotes the operator "set difference".

The expected error-cost functions for sender-driven distributed streaming can be derived in extending [10] to a multiple paths case. Recall that $\epsilon(\pi_l)$ is the probability that data unit l is not delivered on time given all the transmission actions in π_l. Furthermore, the expected cost on path m at transmission opportunity t_i is zero if $a_{mi} = 0$, and otherwise it is equal to the probability that no acknowledgements arrive at sender m by t_i, as a result of previous transmissions of the data unit. Hence, we can write

$$\begin{aligned} \epsilon(\pi_l) &= \prod_{\substack{j<i,\, p: \\ a_{pj}=1}} P\{FTT^p > t_{DTS} - t_j | FTT^p + BTT^m > t_i - t_j\} \\ & \times \prod_{\substack{j \geq i,\, p: \\ a_{pj}=1}} P\{FTT^p > t_{DTS} - t_j\}, \end{aligned} \quad (4)$$

$$\begin{aligned} \rho_m(\pi_l) &= \sum_{\substack{j \geq i: \\ a_{mj}=1}} \prod_{\substack{k<i,\, p: \\ a_{pk}=1}} P\{FTT^p + BTT^m > t_j - t_k | FTT^p + BTT^m > t_i - t_k\} \\ & \times \prod_{\substack{i \leq k < j,\, p: \\ a_{pk}=1}} P\{FTT^p + BTT^m > t_j - t_k\}, \end{aligned} \quad (5)$$

where the conditional probabilities in (4) and (5) can be computed using Bayes' rule [11] and the fact that $P\{x > a, x+y > b\} = P\{x > a\}$, for x, y nonnegative and $a > b$. To this end, note that the probability densities of the sums $FTT^p + BTT^m$ can be obtained as the convolution of the corresponding densities for FTT^p and BTT^m.

C. Rate-Distortion optimization

Based on the expressions for expected distortion, and expected streaming rate as a function of the streaming policy $\boldsymbol{\pi}$, we now have to find the policy vector that minimizes the expected Lagrangian $J(\boldsymbol{\pi})$. This is in general difficult since the terms involving the individual policies π_l in $J(\boldsymbol{\pi})$ are not independent. Therefore, we employ an iterative descent algorithm, called Iterative Sensitivity Adjustment (ISA), in which we minimize the objective function $J(\pi_1, \ldots, \pi_L)$ one variable at a time while keeping the other variables constant, until convergence [9, 10]. The appropriate choice of the Lagrange multipliers λ_m that allow to meet the bandwidth constraints is then determined as follows. We initially select $\lambda_m = \lambda$, for $m = 1, \ldots, M$, and some $\lambda > 0$. We repeatedly re-run the ISA optimization algorithm till convergence, while adjusting one of the Lagrange multipliers every time the optimization algorithm converges. The Lagrange multiplier λ_m is adjusted using the bisection search technique [12–14] till either the target rate is achieved or the interval wherein λ_m can range becomes smaller than a predefined threshold. The procedure outlined above is repeated until we properly adjust all Lagrange multipliers.

It is the duty of sender m, $m = 1, \ldots, M$, to recompute the optimal policy π_l^* for data unit l at every t_i and then execute a_{mi} from π_l^*. This is done in order to account for acknowledgements received by the senders due to packets carrying other data units sent prior to t_i and also to account for prospective bandwidth variations on the network paths. Clearly, the search for the optimal transmission policy, as described before, may lead to systems that does not scale very well in practice; indeed, the overall computational complexity, as well as the computational complexity at each server, increase with the number of senders. This is the price to pay for having a fully distributed streaming strategy, where the sending servers do not communicate among themselves. In the subsequent section, we therefore propose a low complexity algorithm based on source pruning and packet classification that dramatically reduces the computational requirements of the system.

IV. Low-Complexity Distributed Streaming

A. Packet Classification based on Source Pruning

In order to reduce the complexity of the rate-distortion optimization problem described in the previous section, we now design a low-complexity algorithm based on media packet classification. The classification that can be computed offline, is built on a rate-distortion pruning strategy, which has been proposed in [15] for the design of low-complexity adaptive streaming systems. The source pruning strategy defines the set of the most important packets that should be transmitted when the average bit rate is constrained. It proceeds as follows.

We are interested in finding the vector of packet selection actions $\boldsymbol{b} = (b_1, \ldots, b_L)$ for the presentation, where $b_i = 1$ denotes the action of keeping data unit i in the presentation, while $b_i = 0$ signifies the converse. The incurred reconstruction error (or distortion) for the media presentation associated with a particular vector $\boldsymbol{b}$ is denoted $D(\boldsymbol{b})$ and can be computed as

$$D(\boldsymbol{b}) = \sum_{i=1}^{L} \Delta d_i(\mathcal{A}_i(\boldsymbol{b})) \tag{6}$$

where the notation $\mathcal{A}_i(\boldsymbol{b})$ simply signifies the fact that the choice of a subset from $\mathcal{A}_i$ that will be used to reconstruct data unit i depends on the selection vector $\boldsymbol{b}$. Similarly, the associated data rate of the source $R(\boldsymbol{b})$ as a function of the selection vector can be computed as $R(\boldsymbol{b}) = \sum_{i=1}^{L} B_i b_i$. We are interested in finding the optimal selection vector $\boldsymbol{b}^*$ that minimizes the resulting reconstruction error and for which the data rate of the source does not exceed the available resource as given by R^*, i.e.,

$$\boldsymbol{b}^* = \arg\min D(\boldsymbol{b}), \text{ s.t. } R(\boldsymbol{b}) \leq R^* \tag{7}$$

Using the method of Lagrange multipliers the solution to the constrained optimization problem can be replaced with an equivalent convex hull approximation that is obtained as a solution of the unconstrained optimization problem given as

$$\boldsymbol{b}^* = \arg\min D(\boldsymbol{b}) + \lambda R(\boldsymbol{b}), \tag{8}$$

where $\lambda > 0$ is a Lagrange multiplier. The adjustment of λ according to the rate constraint R^* is performed in an iterative fashion using a bisection search technique. Due to interdependencies between media packets, the solution of the optimal policy vector is again computed by the ISA algorithm that minimizes the Lagrangian $J(\boldsymbol{b}) = D(\boldsymbol{b}) + \lambda R(\boldsymbol{b})$ one component at a time, until convergence [9, 10, 15].

The source pruning algorithm leads to a straightforward classification of the media packets as a function of the target streaming rate. Running the pruning scheme at successive target rates directly results in fine packet classification. The packet partitioning then simply performs as follows. Let $R_1, \ldots, R_K$ be a sequence of K monotonically increasing data rates. Packets of a media stream are classified into K sets $S_1, \ldots, S_K$, where the sets S_i are obtained by pruning the source at the corresponding rates R_i, for $i = 1, \ldots, K$. It is important to note that the pruning algorithm typically creates embedded sets[2], i.e., for any two sets S_i and S_j such that $i < j$, it holds that $S_i \subset S_j$. Therefore, a comprehensive pruning of the full stream is not necessary at all target rates, and the classification can be performed iteratively with a limited complexity.

In essence, the procedure described above outlines the operational rate-quality[3] function for the media source, as represented by its data units. This is illustrated in Figure 4, where for each rate point R_i, there is a companion video quality point Q_i, and where the data units are denoted as packets P_l, for $l = 1, \ldots, L$. The incremental increase in quality and rate that each new

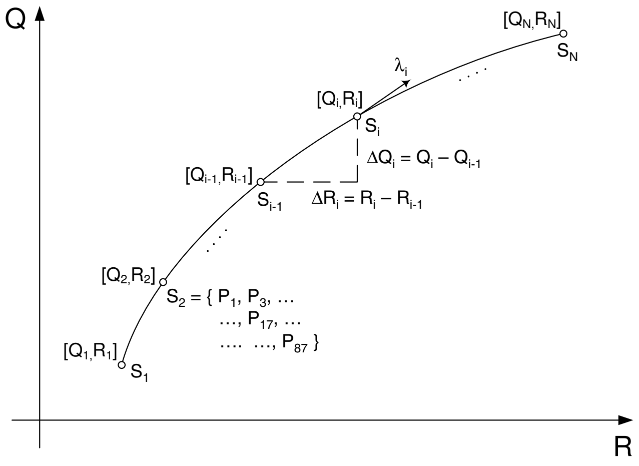

Fig. 4. Operational $R - Q$ function obtained via packet classification.

set S_i provides relative to its predecessor S_{i-1} can be determined as $\Delta Q_i = Q_i - Q_{i-1}, \Delta R_i = R_i - R_{i-1}$[4]. Hence, each

[2]This is due to the fact that the actual rates associated with the sets S_i lie on the convex hull of the operational rate-distortion function for the compressed packetized media [15].

[3]In this context, quality is inversely proportional to reconstruction distortion. For example, distortion can represent the MSE of the reconstructed media units, while quality can then be the PSNR of this quantity.

[4]It can be safely assumed that $R_0 = Q_0 = 0$.

segment i of the operational $R-Q$ function can be characterized with a "gradient" of the function (on that segment) defined as $\lambda_i = \Delta Q_i / \Delta R_i$. In plain words, λ_i denotes the per unit rate increase in quality that the segment i adds to the reconstructed video stream. For more details on the packet pruning algorithm, the reader is referred to the cited reference [15].

B. *Streaming via Packet Partitioning*

We propose the following algorithm for partitioning the packets of a video stream among the M senders based on the available bandwidth. Each sender employs the technique proposed above in order to create a priori the subsets S_i for the video stream. A sufficiently large range of data rates is chosen so that it covers the possible bandwidth fluctuations encountered on the network paths. Moreover, a sufficiently large K is chosen so that there is fine (incremental) division of the data rate range $R_K - R_1$. Note that R_K should be chosen such that it corresponds to the encoding rate of the source, i.e., S_K contains all the packets from the video stream.

Now, given the vector of estimated bandwidth values on every path $\widetilde{R}_1, \ldots, \widetilde{R}_M$ the packet partitioning algorithm proceeds as follows. If there is an $m \in \{1, \ldots, M\}$ such that it holds $\widetilde{R}_m \leq R_K$, then we are done. The video stream is simply streamed on any one of the paths for which the above is true[5]. Otherwise, for $m = 1, \ldots, M$ each sender m solves for the index $\kappa(m)$ according to

$$\kappa(m) = \arg\max_k R_k, \text{ s.t. } R_k \leq \sum_{i=1}^{m} \widetilde{R}_i, \tag{9}$$

and sends to the client the packets from the set $S_{\kappa(m)} \setminus \bigcup_{i=1}^{m-1} S_{\kappa(i)}$, where "$\setminus$" denotes the operator "set difference". In case there is an $m < M$ for which $\kappa(m) = K$, there is no need to run the algorithm further, i.e., for the rest of the indices $m < j \leq M$. In other words, we would reach a point where we could send all the packets from the video stream on the first m paths. Note that employing such a procedure for constructing the partitions of packets sent on each path is possible because of the property that the sets S_i are embedded, as mentioned earlier. We summarize the packet partitioning algorithm in Figure 5.

```
Given $\widetilde{R}_1, \ldots, \widetilde{R}_M$,
  (0)  If $\exists\, m \in \{1, \ldots, M\}$, s.t. $\widetilde{R}_m \leq R_K$
           Send set $S_K$ on path $m$.
           Exit.
  (1)  Else
           Initialize: $m = 1, l(0) = 1$
           while $l(m-1) < K$ and $m \leq K$ do
               $l(m) = \arg\max_k R_k$, s.t. $R_k \leq \sum_{i=1}^{m} \widetilde{R}_i$
               [**Index assignment**]
               Send on path $m$: $S_{l(m)} \setminus \bigcup_{i=1}^{m-1} S_{l(i)}$
               [**Packet partitioning**]
               $m = m + 1$
  (2)  End
```

Fig. 5. Distributed streaming via packet partitioning.

The algorithm can be generalized in a straightforward manner to the case when the network paths exhibit packet loss in the forward direction, in addition to the varying bandwidth. In particular, let ϵ_m denote the erasure rate of packets sent to the client on path m, for $m = 1, \ldots, M$. These quantities can be estimated by the client based on missing sequence numbers of arriving packets and can be piggybacked periodically to the senders on the corresponding acknowledgements. The modification on the senders' part then consists of merely employing the updated bandwidth estimates $\widetilde{R}_1(1-\epsilon_1), \ldots, \widetilde{R}_M(1-\epsilon_M)$ in the packet partitioning algorithm.

Note that partitioning the media packets among the senders corresponds to distributing the streaming load among them based on the available bandwidth on their respective network paths. Hence, the computational complexity of packet scheduling at each server is bounded by the number of packets in the partition allocated to that server. This is equivalent to partitioning the overall search space of transmission policies for each server in the original optimization problem presented in Section III. Therefore, a dramatic reduction of the computational load at each server is achieved relative to the rate-distortion optimal solution.

V. Simulation results

A. *Setup*

This section examines the performance of the proposed optimization framework for distributed streaming of packetized video content. The performance of its low-complexity alternative is also investigated. The video content employed in the simulation

[5]For example, the senders can agree ahead of time on the strategy where the stream is sent on the first (smallest) m for which this condition holds.

experiments are the test video sequences Foreman and Mother & Daughter in QCIF size encoded at 10 fps using an H.264 codec [16]. Each sequence is encoded with a constant quantization level at an average luminance (Y) PSNR of about 36 dB and a Group of Pictures (GOP) size of 20 frames, where each GOP consists of an I frame followed by 19 consecutive P frames. Performance is measured in terms of the average Y-PSNR of the frames of a reconstructed video content at the receiver. Video frames that are not delivered on time are replaced by the receiver using previous frame error concealment.

We consider the case with two streaming servers ($M = 2$) that transmit the video content to a client over two independent network paths. The time interval between transmission opportunities has been set to $T = 100$ ms. It should be noted that at every time instance t_i at which packet scheduling is performed, a sender considers only a subset of the media packets for the presentation. This subset is chosen by employing a sliding window over the whole set of media packets [9], according to the packets' delivery deadlines. In essence, the sliding window selects the packets that are most relevant (pressing) to be sent given the current transmission opportunity t_i.

A simple algorithm is implemented to estimate the transmission bandwidth for the streaming session. The receiving client monitors the forward-trip time (FTT) of arriving packets and piggybacks this information on returning acknowledgement packets to all the sending servers. In particular, let $FTT_1^k, FTT_2^k, \ldots, FTT_P^k$ be the transmission delays experienced by packets received by the client on path k and returned to the servers via the corresponding acknowledgements in the last ΔT seconds. Then, the most recent estimate for the bandwidth (data rate) available on path k is computed by the servers as $\widetilde{R}_k = (1/P)\sum_{j=1}^{P}(B_j/FTT_j^k)$. This is simply the average of the P most recent estimates of the available bandwidth on the path associated with the corresponding received packets, where B_j is the size of packet j in bits (or bytes). The time period employed at the server for bandwidth estimation has been set to $\Delta T = 1$ second, in accordance to the techniques proposed in [17]. It can be noted that other bandwidth estimation methods can be implemented at the server, such as the one proposed in [17] or inspired from [18, 19], since delays and loss rate information are made available via receiver feedbacks. We do not expect that the actual bandwidth estimation method would however change the analysis, and conclusions derived in this paper.

For comparison purposes, in the simulations we also examine the performance of a conventional system, denoted *Baseline*, which performs proportional packet scheduling based only on the available bandwidth values. In particular, the two senders split the packets of a video stream in proportion to the bandwidth estimates on the corresponding network paths. Packet dropping decisions in *Baseline* are performed randomly without taking into account their specific distortion importance. In other words, *Baseline* is distortion-agnostic.

B. *Distributed Streaming over the Internet*

We simulate the behavior of our distributed streaming system in practical settings, and we consider sending the video content over network traces of actual packet delays and losses collected in the Internet. The two senders are located on the East Coast, while the receiving destination is located on the West Coast of the continental USA. We examine the performance of the sending sources for delivering the media content to the receiving destination over their respective network paths using each of the three scheduling mechanisms considered in our experiments. To collect the network traces packet probes of 50 Bytes were sent in both directions on each path every 10ms for the duration of two full days in order to sample continuously the network conditions on the paths. The data rate of the probes (40kbps) represents a small fraction of the access links at the sender and the receiver, thereby not having an effect on the delay and loss characteristics of the paths. For more details on the methodology that was employed to collect the traces and the related issues, the reader is referred to [20].

In Figure 6, we examine the performance of the scheduling schemes in terms of the Y-PSNR quality computed on the sequence reconstructed at the decoder. The performances are reported as a function of the playback delay at the receiver, for streaming the Foreman sequence from two servers. It can be seen that both optimized schemes, *RDOpt* and *PackClas*, outperform the conventional system *Baseline* with a significant margin when the playback delay is small. This is due to the fact that *RDOpt* and *PackClas* can take advantage of the different importance of the media packets when scheduling their transmissions. In particular, by sending the more important packets first and ahead of time, these schemes increase the likelihood of successful delivery of the corresponding data units to the receiving client. This in turn reduces the reconstruction distortion for the media presentation that the client would observe on the average. On the other hand, as the baseline scheme does not have the knowledge of packet importance, it schedules every packet equally, which in turn results in higher average distortion for the reconstructed stream.

It should be mentioned that the lower end of playback delays considered here are sufficiently small (relative to the packet delays experienced on the network paths) to effectively prohibit packet retransmissions. Retransmissions would certainly improve the performance of all three scheduling mechanisms, and especially of the conventional technique. This is evident from Figure 6 for the higher end of playback delay values, when the performances of the three schemes converge. Finally, note that the difference in performance between *RDOpt* and *PackClas* is non-negligible, as shown in Figure 6. This is due to the fact that *RDOpt*, contrarily to *PackClas*, can schedule transmissions of the same packet, in case of high importance, from both senders, which increases the likelihood of its delivery.

We observed similar relative performance between the three systems in the case of Mother & Daughter. As seen from Figure 7, *RDOpt* and *PackClas* again provide substantially improved performance over the baseline scheme for the lower range of playback delays. The performance difference between the three schemes then reduces as the playback delay is increased, which is expected, as argued earlier for the case of the Foreman sequence. Finally, it should be mentioned that the performance differences between

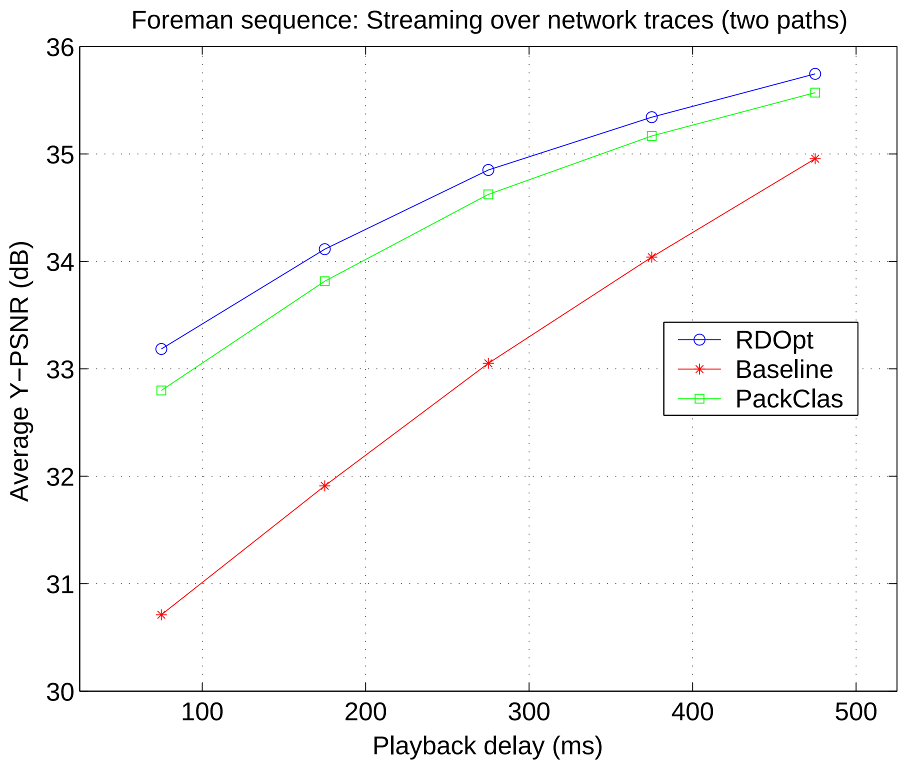

Fig. 6. Distortion performance for streaming Foreman over network traces as a function of play-out delay.

the three systems exhibited in Figure 7 are somewhat smaller relative to the corresponding ones for the Foreman sequence. This is due to the fact that error concealment works more effectively in the case of Mother & Daughter due to the slow-moving nature of this video content. This in turn renders the presence of certain video packets less crucial for the reconstruction quality of the video presentation, thereby reducing the effects of the relative differences in packet delivery that each of the scheduling schemes may contribute to.

C. Performance analysis

Here, we examine in greater detail the performance of the proposed system with a particular emphasis on its adaptivity to bandwidth variations and packet loss. We choose a simple network model in order to facilitate the analysis, where the network bandwidth in the forward direction on each path is randomly varying between a lower and an upper bound. The random fluctuations of the available bandwidth occur every two seconds. The characterization of the bandwidth variations that we employ here has been chosen to match typical bandwidth variations observed in the Internet. At the same time, we do not expect that the performance of the distributed streaming system is highly sensitive to short time variations of the available bandwidth, as long as the play-out delay of the client application allows for buffering delays on the order of the time needed for accurate bandwidth estimation. The playback delay of the client application is set to one second, which is a small values for practical streaming systems. It moreover corresponds to the time period ΔT that is used for bandwidth estimation. In our simple network model, the packet delay densities are assumed to be exponential functions and are inherently tied to the available bandwidth on the network paths. In particular, from the M/M/1 model [21] that is frequently used to model network queues, we know that the mean of the corresponding exponential distribution for the network delay experienced by a packet is $\mu = PackSize/BW$, where $PackSize = 500$ bytes is the average packet size used in our simulations, and BW is the available bandwidth. The range in which the available bandwidth is randomly varying is given as $[BW_{min}, BW_{min} + 20]$ for one of the paths, and $[BW_{min} + 20, BW_{min} + 40]$ for the other network path, where BW_{min} is measured in kbps. Hence, the paths are asymmetric in terms of available bandwidth and the backward channel that is used only for acknowledgments packets is considered to offer sufficient bandwidth in all cases.

1) Bandwidth adaptation: We first analyze the behavior of the distributed streaming system when it faces frequent variations of the available bandwidth. In particular, the packet loss rates on the network paths are assumed to be zero (i.e., $\epsilon_F = \epsilon_B = 0$), and we study how the framework performs in adjusting the streaming rate of the video content to the bandwidth variations of the underlying network. In the simulations, we change BW_{min} across a certain range, and for each of its values we record the corresponding Y-PSNR performances of the rate-distortion optimal system, henceforth denoted *RDOpt*, of the low-complexity technique from Section IV, henceforth denoted *PackClas*, and of the conventional system *Baseline*.

In Figure 8, we show the performances of the three systems under investigation for streaming the Foreman sequence, as a function of the minimum transmission bandwidth available on both paths during a session, i.e., $2 * BW_{min} + 20$. It can be seen

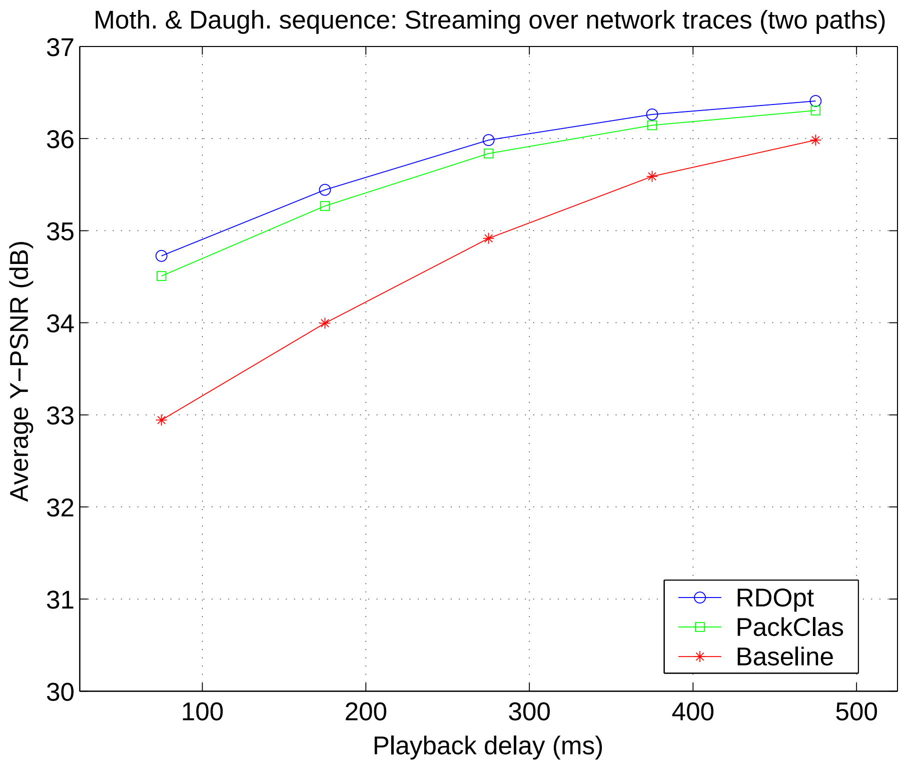

Fig. 7. Distortion performance for streaming Mother & Daughter over network traces as a function of play-out delay.

from the figure that *RDOpt* provides an improved performance over the baseline system almost over the whole range of values considered for the minimum overall bandwidth that is available. The gains in performance are especially significant in the lower half of the bandwidth range. For example, at 60 kbps minimum overall bandwidth there is a difference of 4.5 dB in performance between the two systems. The improvement is due to the fact that the optimized system takes into account the distortion importance of the individual packets while adapting the video stream to the available bandwidth. In particular, by selecting the most important packets for transmission for the given available data rate on each path, *RDOpt* ensures the best possible reconstruction quality of the video presentation at the receiver. On the other hand, *Baseline* performs bandwidth adaptation without treating the various packets preferentially, as it is distortion-agnostic. Therefore, some more important packets may be dropped at the expense of others, less important ones, which would ultimately lead to a degradation in video quality at the client.

The low-complexity technique also outperforms the baseline system, as shown in Figure 8. This is quite encouraging, as both systems only induce a small online complexity. The improved performance of *PackClas* is due to the fact that the packet partitions from which the senders stream the video data are selected based on the subsets of packets S_i. These in turn are selected such that they correspond to the maximum possible video quality for the corresponding available data rates, as explained in Section IV. It should be noted though that the low-complexity technique provides a somewhat degraded performance relative to the global optimization framework. The difference in performance between the two systems reaches up to 1-1.2 dB in the lower end of data rates.This is anticipated, as in the former system streaming is performed based on the packet subsets S_i that are selected ahead of time. Therefore, they provide less flexibility in terms of adapting to dynamic bandwidth variations relative to the optimization framework, in which at every instance a sender has access to all the packets from a video stream when making transmission decisions. This is the necessary price that systems like *PackClas* pay in order to reduce their online complexity by having pre-defined sets of packets to choose from when streaming.The number of packet sets is finite, and therefore fine adaptation to bandwidth variations is constrained by the granularity of the K target rates $[R_1, R_2, ..., R_K]$ used for the packet partitioning.

Finally, it can be seen from Figure 8 that all three systems perform alike for sufficiently large minimum overall bandwidth. This is expected here since there is sufficient bandwidth available throughout the session to ensure timely delivery of all packets to the receiver in the case of each system. In other words, no packet needs to be dropped anymore due to a mismatch between the network bandwidth variations and the dynamically varying source encoding rate.

Similar relative performances for the three systems are observed for streaming Mother & Daughter, as shown in Figure 9. It can be seen that again both *RDOpt* and *PackClass* outperform *Baseline* almost over the whole range of bandwidth values under consideration, with gains reaching up to 6-8 dB in the lower half of the bandwidth range. Similarly, there is a performance gap between *RDOpt* and *PackClass*, which is due to the reason explained earlier. Finally, when the overall available bandwidth reaches a value at which no bandwidth adaptation is needed throughout the session, all three systems perform identically, as illustrated by

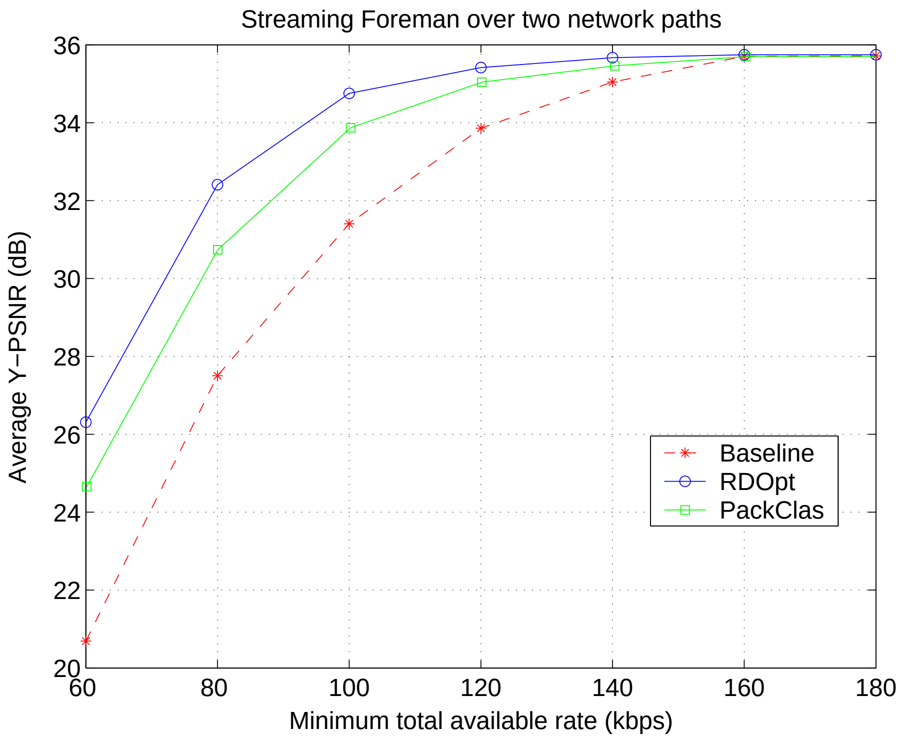

Fig. 8. Rate-distortion performance for bandwidth adaptation of Foreman.

their performances for 80 kbps minimum overall bandwidth.

2) Adaptation to bandwidth and packet loss: Next, we examine the performance of *RDOpt*, *PackClas*, and *Baseline* when adapting to both bandwidth variations and random packet loss. In particular, in addition to dynamic bandwidth variations the communication channels also exhibit random packet erasures. Therefore, packets will be lost during transmission and a streaming system needs to decide then whether it would retransmit packets or send new packets. This trade-off is necessitated by the fact that the available bandwidth is insufficient to support sending all the packets together, including the (re)transmissions. Note that the target rates used for the packet classification in *PackClas* are adapted to match the effective bandwidth on the streaming paths (see Section IV-B), in order to take the loss process into account. In the new set of simulations, we measure the performances of the three systems as a function of the available data rate, but now in the presence of packet loss. Specifically, we examine packet loss rates (ϵ_F) of 5% and 10% on the forward channels from the senders to the receiver.

In Figure 10, we show the performances of the three distributed streaming systems for transmitting the Foreman sequence on lossy channels. First, it can be seen that all of them exhibit a degraded performance relative to the corresponding results shown in Figure 8, where only bandwidth adaptation is performed. This is expected since each of the streaming systems has to take into account retransmissions of lost packets, in addition to discarding packets due to bandwidth variations. Therefore, higher data rates on the communication channels are needed to achieve the same Y-PSNR performance relative to the case of bandwidth adaptation only. Second, it can also be seen that the performances of *RDOpt*, *PackClas*, and *Baseline* degrade with increasing the packet loss rate. For example, for $\epsilon_F = 5\%$, all three systems exhibit a Y-PSNR performance within the 33-34 dB range when the total available data rate on the channels is 140 kbps. However, that performance reduces to being in the range of 30-32 dB for the same data rate at packet loss rate of 10%. Such a performance behavior across the three systems is also expected, as increasing the loss rate reduces the number of packets that can be delivered on time to the receiver, given a fixed play-out delay.

The two optimized systems *RDOpt* and *PackClas* provide the most significant gains over *Baseline* in the lower end of data rates under consideration, especially for $\epsilon_F = 5\%$, as seen from Figure 10 (left). This is expected as these systems can trade off the importance of each packet for the effective data rate, while the conventional system is distortion agnostic. However, as the packet loss rate is increased *RDOpt* and *PackClas* cannot take a lot of advantage of their knowledge of the packets' distortion importance. Even though packets are prioritized in terms of transmission, they become more likely to be lost and there is not enough data rate to perform loss recovery by retransmission. On the other hand, when there is sufficient data rate available on the channels, all three systems can deal effectively with packet losses by retransmission, as illustrated by their similar performance in the upper end of the data rate range. Note that again *RDOpt* outperforms *PackClas* over all data rates and packet loss rates considered, as shown in Figure 8. *RDOpt* achieves that by taking advantage of the fact that it considers for transmission the complete set of video packets, all of the time. This in turn allows for more efficient dynamic adaption, as explained before.

In Figure 11 we show the corresponding performances of *RDOpt*, *PackClas*, and *Baseline* for streaming the 'Mother & Daugh-

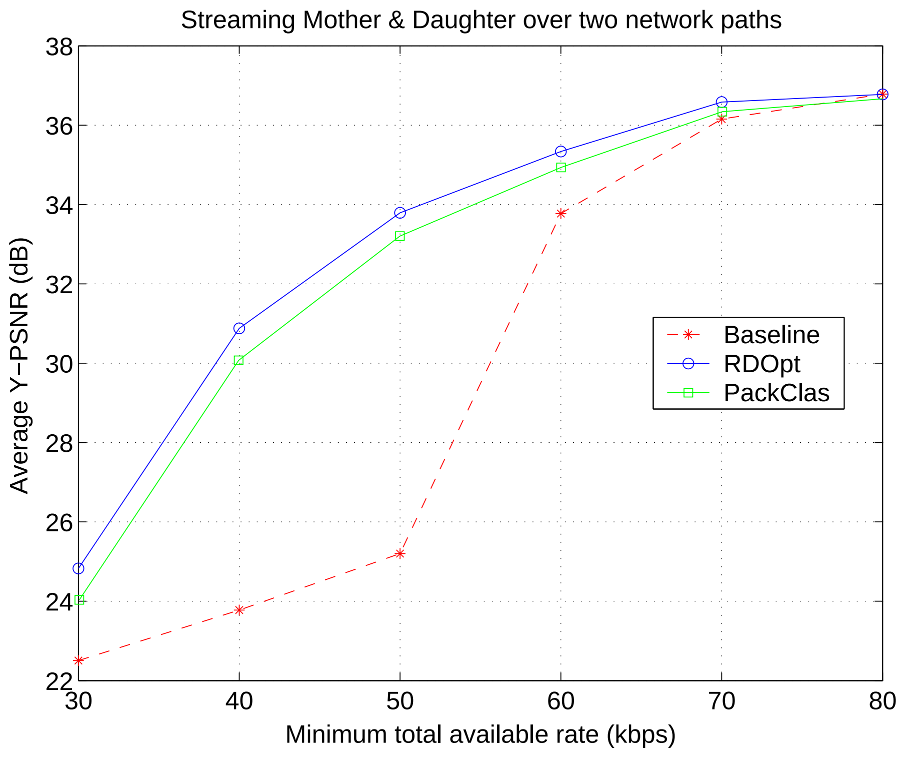

Fig. 9. Rate-distortion performance for bandwidth adaptation of Mother & Daughter.

ter' sequence. It can be seen that the relative performances of the three systems across the different data rates and packet loss rates under consideration exhibit a similar behavior to those for the case of the Foreman sequence. In particular, the optimized systems outperform more significantly the conventional system in the lower end of data rates and especially for lower packet loss rates. As the packet loss rate increases the performances of the three systems degrade significantly and therefore become more alike. In the same spirit, as the available data rate is sufficiently large, the three systems perform alike again. In this case, there is enough bandwidth to recover (almost completely) from packet losses by retransmissions, as shown in Figure 11.

3) Bandwidth estimation performance: Finally, we examine the performance of the simple algorithm described earlier for estimating the available bandwidth on a network path. In Figure 12 (left), we show the bandwidth evolution over time on one of the network paths when streaming Foreman. The solid line denotes the actual bandwidth values, while the dashed line represents its estimate obtained using the proposed algorithm. It can be seen from Figure 12 (left) that the estimated values track quite well the dynamic variations of the actual bandwidth on the path, especially given the fact that random bandwidth changes are initiated frequently (every two seconds) relative to the bandwidth estimation period (every one second). Notable discrepancies between the actual and the estimated values occur only when there is a substantial sudden reduction in the available bandwidth, as illustrated at a few points in the bandwidth trace shown in Figure 12 (left) (for example around Time equal to 18s and 60s on the x-axis). This is expected, as the procedure for estimating the available bandwidth is causal, i.e., it is based on previously received acknowledgement packets within a time period. Therefore, when the available network bandwidth suddenly and substantially drops there are not so many acknowledgement packets received within the next time period that can be used for accurate bandwidth estimation.

In order to examine how the frequency of bandwidth variations affects the accuracy of the estimation technique, we performed the same simulations as before, but now with random bandwidth fluctuations occurring every five seconds. These results are shown in Figure 12 (right). It can be seen that the estimated bandwidth values track even closer now the actual bandwidth on a network path. That is because the bandwidth changes less frequently now relative to the estimation period, which allows for better and more stable estimates to be performed using the proposed technique, as illustrated in Figure 12 (right).

Finally, a useful evidence for the interpretation of the streaming results presented earlier is the source encoding rate of the Foreman video sequence used in the simulations, which has an average value of 82.23 kbps. Even though the minimum overall bandwidth is 100 kbps, and the average source encoding rate is almost 20 kbps smaller than that, still there are points along the time axis when the instantaneous source encoding rate exceeds the available network bandwidth (marked with arrows in Figure 13 at Time roughly equal to 8s and 35s on the x-axis). Therefore, packets need to be dropped at these points to compensate for the insufficient transmission bandwidth, which results in loss in Y-PSNR performance. It is interesting to note that the same situation occurs when the instantaneous source rate is larger than the estimate of the available network bandwidth (though not necessarily of its actual value), as evident from Figure 13 at Time equal to roughly 22s. This is exactly where the packet prioritization performed

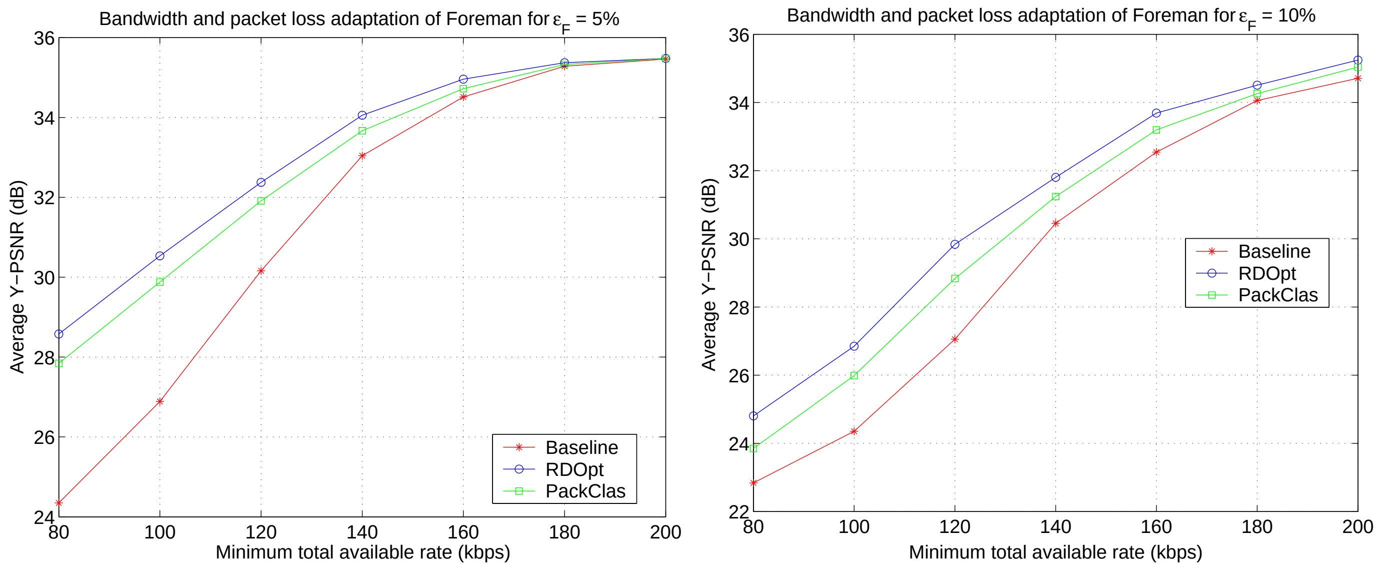

Fig. 10. Bandwidth and packet loss adaptation of Foreman: (left) $\epsilon_F = 5\%$ and (right) $\epsilon_F = 10\%$.

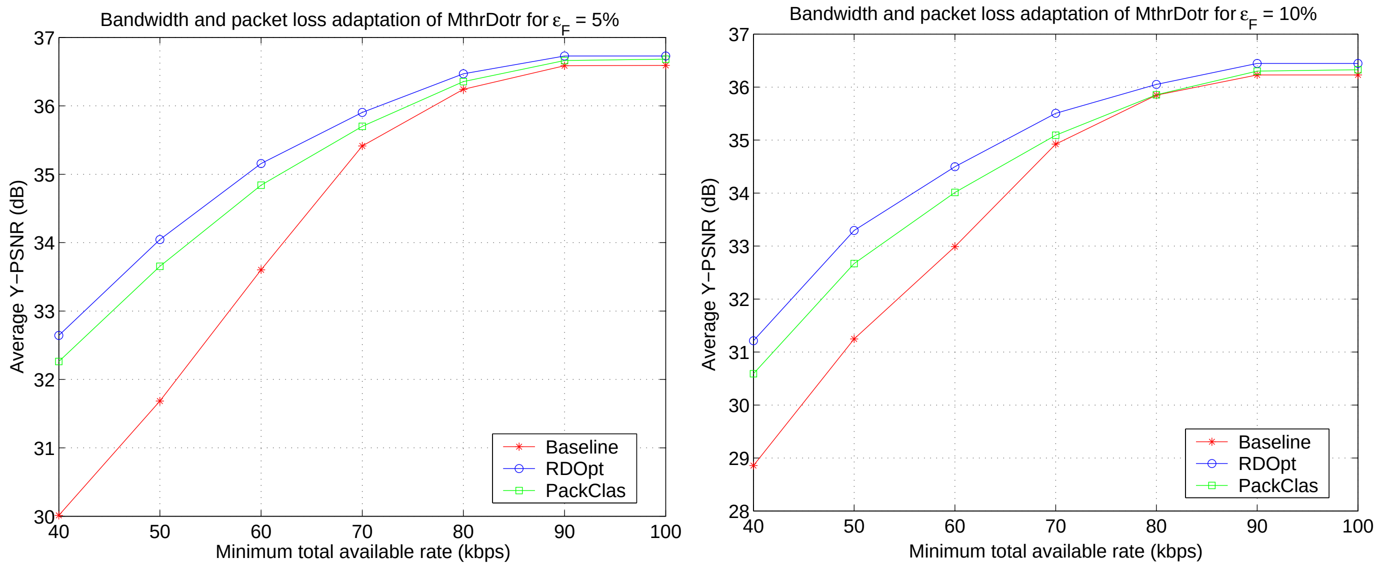

Fig. 11. Bandwidth and packet loss adaptation of Mother & Daughter: (left) $\epsilon_F = 5\%$ and (right) $\epsilon_F = 10\%$.

by the optimized distributed streaming solutions proves to be advantageous compared to a distortion-agnostic schedulers. Lastly, it should be pointed out that analogous observations regarding the dynamics of the source rate and the available network bandwidth were made for the case of streaming Mother & Daughter.

D. *Discussion*

We have shown that the distributed streaming system based on packet classification provides performance that is quite competitive with the rate-distortion optimal solution. The suboptimal behavior is due to several design choices, which objectives are mainly driven by reducing the computational complexity of the optimal streaming solution. In particular, a priori partitioning of the media packets breaks the dependencies between media units, since each server manages a distinct subset of data units, independently of the streaming strategies decided by other servers, on other complementary subsets of data units. Additionally, we observed in our simulations that the rate-distortion optimal strategy sometimes sends the same packet from different servers in order to increase the probability of on-time arrival at the client of important packets. Such a strategy is excluded from the streaming policies available in *PackClas* scheme, due to the partitioning of the media packets into distinct subsets. Finally, another limitation of *PackClas* stems from the granularity of target rates when classification of the media packet is performed originally (see Section IV-B). For computational complexity reasons, the granularity cannot be chosen arbitrarily fine, which in turn affects how efficient *PackClas* can be when adapting to bandwidth variations. Nonetheless, *PackClas* proves to be a competitive solution for

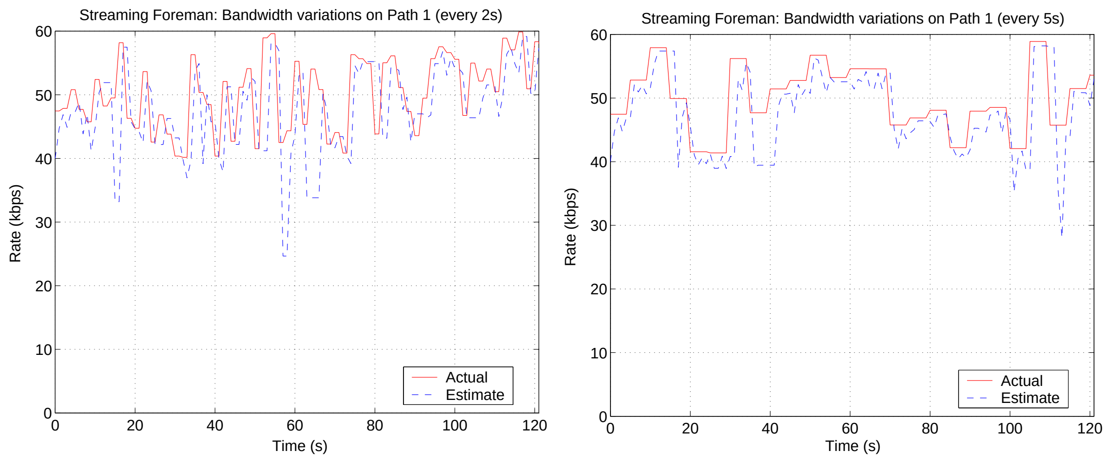

Fig. 12. Bandwidth variations over time on a network path when streaming Foreman. Random bandwidth changes occur every (left) 2s, (right) 5s.

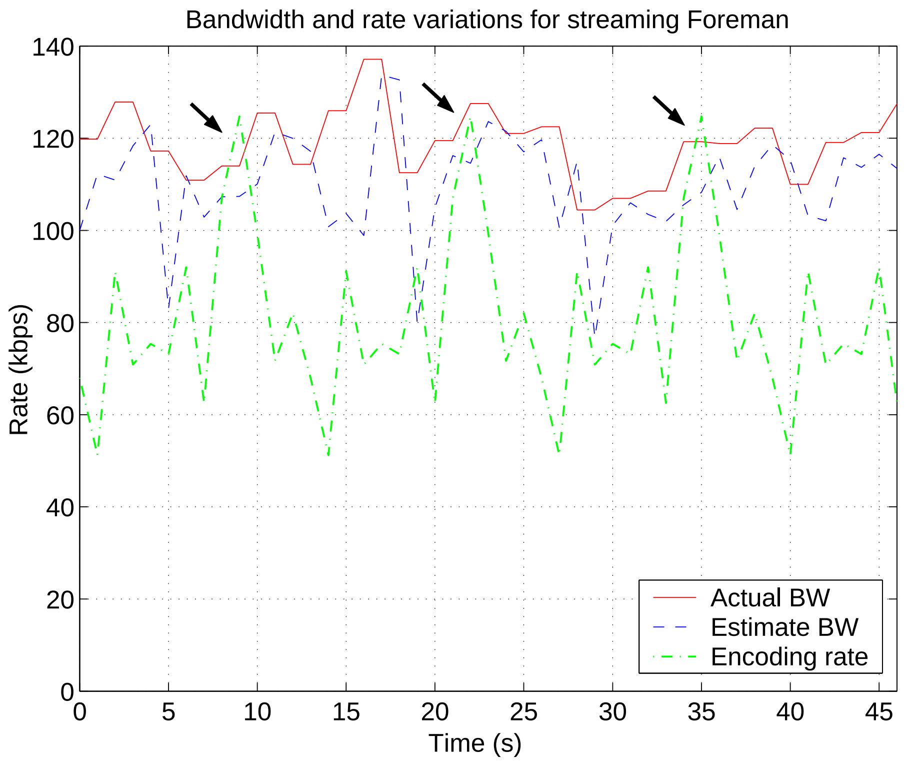

Fig. 13. Dynamic variations of overall bandwidth on the paths and of encoding rate for Foreman.

distributed streaming applications thanks to its low computational complexity, ease of deployment due to minimal requirements on the receiver, and yet efficient rate-distortion performance.

VI. Conclusions

We presented a system for rate-distortion optimized packet scheduling in distributed video streaming scenarios where several sources collaborate to serve a media client. The system consists of an optimization framework for scheduling the packet transmissions at the individual senders. Using bandwidth estimates based on client feedbacks, the senders can independently, but still in coordination, decide what the most important packets are to transmit on every network path, for the given bandwidth estimates. We designed a low-complexity solution that pre-computes optimized packet schedules ahead of time thereby reducing substantially the required online complexity during streaming. This is achieved by employing packet classification, via source pruning, of the compressed video stream at different data rates. Extensive simulations demonstrate minimal performance loss compared to the complex rate-distortion optimal solution, as well as significant performance gains over a conventional distortion-agnostic system for distributed streaming with comparable online complexity.

VII. Acknowledgments

The authors would like to thank Dr A. Markopoulou for providing the Internet traces [20] used in the simulations.

References

[1] J. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in paralel: using tornado codes to speed up downloads," in *Proc. Conf. on Computer Communications (INFOCOM)*, vol. 1. New York City, USA: IEEE, Mar. 1999, pp. 275–283.

[2] T. Nguyen and A. Zakhor, "Distributed video streaming over internet," in *Proc. Multimedia Computing and Networking*, vol. 4673. San Jose, CA: SPIE, Jan. 2002, pp. 186–195.

[3] ——, "Distributed video streaming with forward error correction," in *Proc. Int'l Packet Video Workshop*, Pittsburg, PA, Apr. 2002.

[4] A. Majumdar, R. Puri, and K. Ramchandran, "Distributed multimedia transmission from multiple servers," in *Proc. Int'l Conf. Image Processing*, vol. 3. Rochester, NY, USA: IEEE, Sept. 2002, pp. 177–180.

[5] J. Kim, R. M. Mersereau, and Y. Altunbasak, "Network-adaptive video streaming using multiple description coding and path diversity," in *Proc. Int'l Conf. Multimedia and Exhibition*, vol. 2. Baltimore, MD, USA: IEEE, July 2003, pp. 653–656.

[6] J. Chakareski and B. Girod, "Server diversity in rate-distortion optimized streaming of multimedia," in *Proc. Int'l Conf. Image Processing*, vol. 3. Barcelona, Spain: IEEE, Sept. 2003, pp. 645–648.

[7] J. Apostolopoulos, T. Wong, W.-T. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proc. Infocom*, vol. 3. New York City, NY, USA: IEEE, June 2002, pp. 1736–1745.

[8] J. Apostolopoulos, W.-T. Tan, and S. Wee, "Performance of a multiple description streaming media content delivery network," in *Proc. Int'l Conf. Image Processing*, vol. 2. Rochester, NY, USA: IEEE, Sept. 2002, pp. 189–192.

[9] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2001-35, Feb. 2001.

[10] J. Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *Proc. Data Compression Conference*. Snowbird, UT: IEEE Computer Society, Mar. 2003, pp. 203–212.

[11] A. Papoulis and S. Unnikrishna Pillai, *Probability, Random Variables and Stochastic Processes*. New York: McGraw-Hill Inc., 2001, 4-th ed.

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

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

[14] G. M. Schuster, G. Melnikov, and A. K. Katsaggelos, "A review of the minimum maximum criterion for optimal bit allocation among dependent quantizers," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 3–17, Mar. 1999.

[15] J. Chakareski and P. Frossard, "Low-complexity adaptive streaming via optimized a priori media pruning," in *Proc. Workshop on Multimedia Signal Processing*. Shanghai, China: IEEE, Oct./Nov. 2005.

[16] Telecom. Standardization Sector of ITU, "Video coding for low bitrate communication," *Draft ITU-T Recommendation H.264*, Mar. 2003.

[17] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proc. 4^{th} Passive and Active Measurements Workshop*, La Jolla, CA, Apr. 2003.

[18] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. Data Communication, Ann. Conf. Series*. Stockholm, Sweden: ACM, Aug. 2000, pp. 43–56.

[19] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A TCP-friendly rate adjustment protocol for continuous media flows over best effort networks," in *Proc. Int'l Conf. on Measurement and Modeling of Computer Systems*, vol. 27. Atlanta, USA: ACM, June 1999, pp. 220–221.

[20] A. Markopoulou, F. Tobagi, and M. Karam, "Assessing the quality of voice communications over Internet backbones," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 747–760, Oct. 2003.

[21] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, 1992.