# Rate-Distortion Optimized Streaming from the Edge of the Network

Jacob Chakareski*, Philip A. Chou† and Bernd Girod*

*Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA
Telephone: (650) 724-3628, Fax: (650) 724-3648, Email: {cakarz,bgirod}@stanford.edu

†Microsoft Corporation, Redmond, WA 98052-6399 USA
Telephone: (425) 706-3869, Fax: (425) 936-7329, Email: pachou@microsoft.com

***Abstract*—This paper addresses the problem of streaming packetized media over a lossy packet network through an intermediate proxy server to a client, in a rate-distortion optimized way. The proxy, located at the junction of the backbone network and the last hop to the client, coordinates the communication between the media server and the client using hybrid receiver/sender-driven streaming in a rate-distortion optimization framework. The framework enables the proxy to determine at every instant which packets, if any, it should either request from the media server or retransmit directly to the client, in order to meet a constraint on the average transmission rate on the last hop while minimizing the average end-to-end distortion. Performance gains of up to 2 dB and up to 4 dB are observed over rate-distortion optimized sender-driven systems for the case when the last hop is wireline and wireless, respectively.**

## I. Introduction

We consider the problem of streaming packetized media over a lossy backbone packet network through a proxy server to a client, using a hybrid receiver/sender driven transmission scheme. The proxy is located at the junction of the backbone network and the last hop to the client, which could be wireline or wireless. Packets may be lost in the backbone network due to congestion, or in the last hop due to erasures. In the case when the last hop is wireless, packet payloads might also be corrupted with a non-zero bit error rate (BER) due to fading. The scenario under consideration is illustrated in Figure 1.

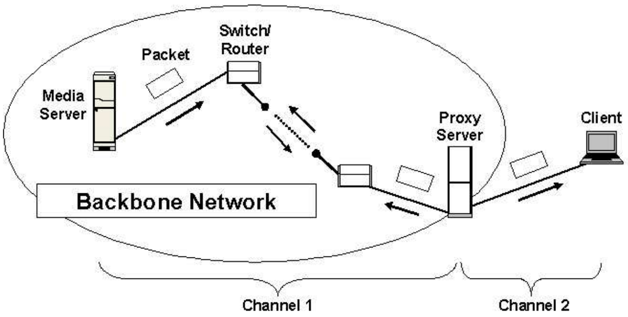

Fig. 1. Streaming on demand to a client through a proxy server.

It is assumed that the server, the proxy and the client do not have control over any network resources available along the communication path between them. For example, a sender cannot change the delivery priorities placed on its packets by the backbone network, the rate of the channel code applied to packets sent across a wireless last hop, etc.

We introduce a novel streaming system, centered around a proxy server equiped with a packet scheduling optimization procedure. The proxy employs a hybrid receiver/sender driven transmission scheme to communicate with the media server and with the client. The optimization procedure is based on the Iterative Sensitivity Adjustment (ISA) algorithm introduced in [1], modified here to suit the settings of the problem under consideration. Using the optimization procedure, rather than streaming the packetized media in a fixed sequence according to presentation time, the proxy can choose a transmission policy for each data unit that minimizes the expected end-to-end distortion of the entire presentation subject to a transmission rate constraint. The solution to this resource allocation problem is obtained by minimizing a Lagrangian, taking into account the data units' dependence relationships in addition to their different delivery deadlines and basic importances.

To our knowledge, the most closely related contemporaneous works are [1–4] where the authors have studied distortion-rate optimized streaming over lossy packet networks to wireline and to wireless clients, in both sender-driven and receiver-driven scenarios, and [5] which considers proxy caching in a cost-distortion optimization framework. Another related work is [6] where the media server exploits feedback from a proxy to better adapt the media content sent to the client.

## II. Preliminaries

In a streaming media system, the encoded data are packetized into *data units* and are stored in a file on a media server. All of the data units in the presentation have interdependencies, which can be expressed by a directed acyclic graph as illustrated in Figure 2. Each node of the graph corresponds to a data unit, and each 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 $t_{DTS,l}$, and an importance $\Delta D_l$. The size $B_l$ is the size of the data unit in bytes. The decoding time $t_{DTS,l}$ is the time

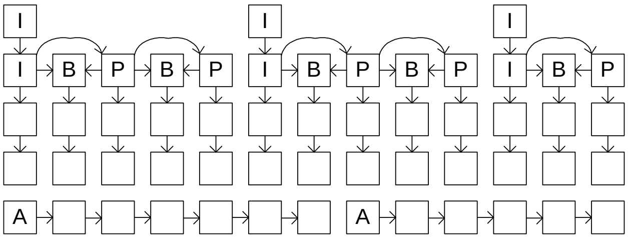

Fig. 2. Typical directed acyclic dependency graph for video and audio data units.

at which the decoder is scheduled to extract the data unit from its input buffer and decode it. (This is the decoder timestamp, in MPEG terminology.) Thus $t_{DTS,l}$ is the *delivery deadline* by which data unit $l$ must arrive at the client, or be too late to be usefully decoded. Packets containing data units that arrive after the data units' delivery deadlines are discarded. The importance $\Delta D_l$ is the amount by which the distortion at the client will *decrease* if the data unit arrives on time at the client and is decoded.

In the following with Channel 1 we denote the network path between the media server and the proxy and with Channel 2 the last hop (see Figure 1). Channel 1 is modeled as an independent time-invariant packet erasure channel with random delays. This means that if the media server inserts a packet into the network at time $t$, then the packet is lost with some probability, say $\epsilon_{F_1}$, independent of $t$. However, if the packet is not lost, then it arrives at the proxy server at time $t'$, where the forward trip time $FTT_1 = t' - t$ is randomly drawn according to probability density $p_{F_1}$. The backward channel is similarly characterized by the probability of packet loss $\epsilon_{B_1}$ and delay density $p_{B_1}$. Furthermore Channel 2 is modeled as an independent time-invariant packet erasure channel with random delays at the packet level and as a binary symmetric channel (BSC) at the bit level [3, 4]. Hence the forward channel is characterized with random loss, delay density and BER $\epsilon_{F_2}, p_{F_2}, BER_{F_2}$. These induce a modified probability of packet loss, $\epsilon'_{F_2}$ which also accounts for packets being discarded by the client platform due to corrupted IP/UDP/RTP header, i.e. $\epsilon'_{F_2} = \epsilon_{F_2} + (1 - \epsilon_{F_2})(1 - (1 - BER_{F_2})^{N_h})$, where $N_h$ is the packet header size in bits. Furthermore, even if a packet arrives at the client application, its payload may still be corrupted with probability $\epsilon_P = 1 - (1 - BER_{F_2})^{N_p}$, where $N_p$ is the size of the payload in bits. Similarly the backward channel is characterized with $\epsilon_{B_2}, p_{B_2}, BER_{F_2}$ which in turn induce a modified probability of packet loss $\epsilon'_{B_2}$.

## III. Hybrid Receiver/Sender Driven Transmission

A multimedia session starts when a client requests a presentation from the media server. The request packet is received by the proxy server and is not forwarded further. The proxy then sends a request to the media server for a rate-distortion preamble for the desired presentation. The preamble contains in effect the directed acyclic dependency graph for the presentation as well as $B_l$, $t_{DTS,l}$, and $\Delta D_l$ for each data unit $l$.

After it has the preamble, the proxy starts communicating with both the media server and the client using a hybrid receiver/sender driven transmission scheme. The communication with the media server is receiver-driven, where the proxy sends request packets to the media server, requesting a particular data unit to be transmitted to the client. The media server responds to a request, by sending a packet with the requested data unit to the client. The proxy may transmit request packets for a particular data unit periodically if necessary, until it observes a returning packet for the first time, or until the proxy gives up requesting transmission of that data unit. The operation of the proxy differs slightly depending on the nature of Channel 2, after a returning packet for a data unit has been observed for the first time.

If Channel 2 is wireline, the proxy stores a copy of the first returning packet in its buffer for later retransmissions and stops requesting that data unit from the media server. If Channel 2 is wireless, the proxy augments the payload of every returning packet with a Cyclic Redundancy Check (CRC) code before forwarding it to the client. This enables the client to verify the integrity of the data unit. Similarly to the wireline case, upon seeing a returning packet for the first time, the proxy stores the CRC augmented data into its buffer for later retransmissions and stops requesting this data unit from the media server. Using the terminology of [3, 4] we denote the CRC augmented data unit a *systematic* packet.

Once the proxy has a copy of a data unit in its buffer, it can then start communicating the data unit directly to the client using a sender-driven transmission scheme. For the case when Channel 2 is wireline, the client responds with a positive acknowledgement (ACK) to an incoming packet. Here the proxy may retransmit the buffered data unit periodically if necessary, until it observes a returning acknowledgement from the client for that data unit, or until the proxy server gives up transmitting the data unit. For the case when Channel 2 is wireless, the client can respond with a positive or negative (NAK) acknowledgement to an incoming packet. We assume here that the client may observe bit errors in the packet payload. That is, we assume that the transport layer runs a protocol such as UDP Lite [7], in which the transport checksum is applied only to the header, rather than to both the header and payload. In this case the proxy employs a hybrid FEC/ARQ error control scheme denoted Incremental Redundancy (IR) transmission [3, 4] that operates as follows. The proxy may transmit systematic packets periodically if necessary, until it receives a positive or negative acknowledgement from the client or until the proxy gives up transmitting the data unit. Once the proxy receives a NAK, it may begin to transmit parity packets periodically if necessary, until it receives an ACK or until it gives up. The proxy generates the parity packets by applying a recursive systematic convolutional (RSC) code to different pseudo-random interleavings of the buffered systematic packet. The client attempts to decode each packet that it receives, using the CRC for systematic packets, or using the List Viterbi algorithm [8] for the first parity packet, or using the Turbo decoding algorithm [9] for subsequent parity packets. The client immediately transmits an ACK packet to the proxy upon successful decoding, or a NAK packet upon decoding failure. The header of the acknowledgement packet identifies the data packet being positively or negatively acknowledged; the payload of the acknowledgement packet is empty. For more details on the IR transmission and its performance we refer the reader to [4].

## IV. R-D Optimization using Sensitivity Adjustment

Suppose there are $L$ data units in the multimedia session. Let $\pi_l$ 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. A transmission policy $\pi_l$ is a schedule according to which the proxy sends request packets to the server or sends systematic or parity packets to the client, for data unit $l$.

Any given policy vector $\boldsymbol{\pi}$ induces an expected distortion $D(\boldsymbol{\pi})$ and an expected transmission rate $R(\boldsymbol{\pi})$ for the multimedia session. We seek the policy vector $\boldsymbol{\pi}$ that minimizes the Lagrangian $D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi})$ for some Lagrange multiplier $\lambda > 0$, and thus achieves a point on the lower convex hull of the set of all achievable distortion-rate pairs.

The expected transmission rate $R(\boldsymbol{\pi})$ is the sum of the expected transmission rates for each data unit $l \in \{1, \ldots, L\}$:

$$R(\boldsymbol{\pi}) = \sum_l B_l \rho(\pi_l). \tag{1}$$

where $B_l$ is the number of bytes in data unit $l$ and $\rho(\pi_l)$ is the *expected cost* per byte, or the expected number of transmitted bytes per source byte under policy $\pi_l$. 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 $\epsilon(\pi_l)$ for $l \in \{1, \ldots, L\}$ that data unit $l$ does not arrive at the receiver on time under policy $\pi_l$:

$$D(\boldsymbol{\pi}) = D_0 - \sum_l \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})), \tag{2}$$

where $D_0$ is the expected reconstruction error for the presentation if no data units are received and $\Delta D_l \prod_{l' \preceq l}(1 - \epsilon(\pi_{l'}))$ is the expected reduction in reconstruction error due to data unit $l$. The product $\prod_{l' \preceq l}(1 - \epsilon(\pi_{l'}))$ is the probability that data unit $l$ and all of its ancestors in the acyclic directed graph (see Figure 2) arrive at the receiver on time under their respective policies.

Finding a policy vector $\boldsymbol{\pi}$ that minimizes the expected Lagrangian $J(\boldsymbol{\pi}) = D(\boldsymbol{\pi}) + \lambda R(\boldsymbol{\pi})$, for $\lambda > 0$, is difficult since the terms involving the individual policies $\pi_l$ in $J(\boldsymbol{\pi})$ are not independent. Therefore, we employ an iterative descent algorithm, called the SA algorithm, in which we minimize the objective function $J(\pi_1, \ldots, \pi_L)$ one component at a time while keeping the other variables constant, until convergence. For more details on the optimization procedure, the reader is referred to [10].

## V. Experimental Results

Here we investigate the end-to-end distortion-rate performance for streaming one minute of packetized audio content using different methods. The audio content, the first minute of Sarah McLachlan's *Building a Mystery*, is compressed using a scalable version of the Windows Media Audio codec. The codec produces a group of twelve 500-byte data units every 0.75 seconds for a maximum data rate of 64 Kbps. All twelve data units in the $m^{th}$ group receive the same decoding timestamp, equal to 0.75*m*. The playback delay is 750 ms.

Channel 1 is specified with the following parameters: $\epsilon_{F_1} = \epsilon_{B_1} = 10$ %, $\kappa_{F_1} = \kappa_{B_1} = 50$ ms, $\eta_{F_1} = \eta_{B_1} = 2$ nodes, $1/\alpha_{F_1} = 1/\alpha_{B_1} = 25$ ms. Transmitted packets on this channel are dropped at random, with a drop rate $\epsilon_{F_1}$ or $\epsilon_{B_1}$. Those packets that are not dropped receive a random delay, where for the delay density $p_{F_1/B_1}$ we use the shifted Gamma distribution with parameters $(n_{F_1/B_1}, \alpha_{F_1/B_1})$ and right shift $\kappa_{F_1/B_1}$.

In the following we compare the performance of two streaming systems in the two possible scenarios for the last hop. *Sender-driven* is a system that performs R-D optimized scheduling of the packet transmissions at the media server. *Proxy-driven* is the system presented in this work, which also performs R-D optimized scheduling of the packet transmissions but at the proxy server. The Lagrange multiplier $\lambda$ is fixed for the entire presentation for both systems. We examine the signal-to-noise ratio (SNR) in dB of the end-to-end perceptual distortion, averaged over the one minute long audio clip, as a function of the available bitrate (Kbps).

*1) Wireline Last Hop:* Channel 2 is specified here with the following parameters: $\epsilon_{F_2} = \epsilon_{B_2} = 1$ %, $\kappa_{F_2} = \kappa_{B_2} = 5$ ms, $\eta_{F_2} = \eta_{B_2} = 1$ node, $1/\alpha_{F_1} = 1/\alpha_{B_1} = 5$ ms, $BER_{F_2} = BER_{B_2} = 0$. Transmitted packets on this channel are dropped at random, with a drop rate $\epsilon_{F_2}$ or $\epsilon_{B_2}$. Those packets that are not dropped experience a random delay, where for the delay density $p_{F_2/B_2}$ we use the shifted Gamma distribution with parameters $(n_{F_2/B_2}, \alpha_{F_2/B_2})$ and right shift $\kappa_{F_2/B_2}$.

For this scenario *Sender-driven* is the system introduced in [1]. It can be seen from Figure 3 that *Proxy-driven* performs consistently better that *Sender-driven* over all available bit rates. The gains in performance reach up to 2 dB and decrease as the bit rate decreases.

*2) Wireless Last Hop:* In this scenario Channel 2 is specified with the following parameters: $\epsilon_{F_2} = \epsilon_{B_2} = 0$ %, $p_{F_2/B_2} = \delta(t - \tau)$ with delay $\tau = 5$ ms. A range of bit error rates is examined $BER_{F_2} = BER_{B_2} \in \{10^{-6}, ..., 10^{-1}\}$. Packets transmitted on the wireless link are dropped randomly with a drop rate $\epsilon'_{F_2}$ or $\epsilon'_{B_2}$ introduced in Section II. Those packets that are not dropped receive a constant delay $\tau$ and have their payload corrupted with the given BER of the channel, using a pseudo-random number generator. The pseudo-random number

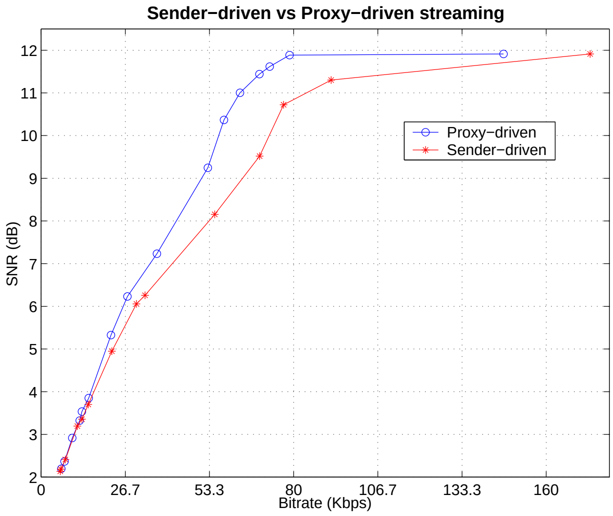

Fig. 3. RD performance of Sender-driven and Proxy-driven streaming for a wireline last hop

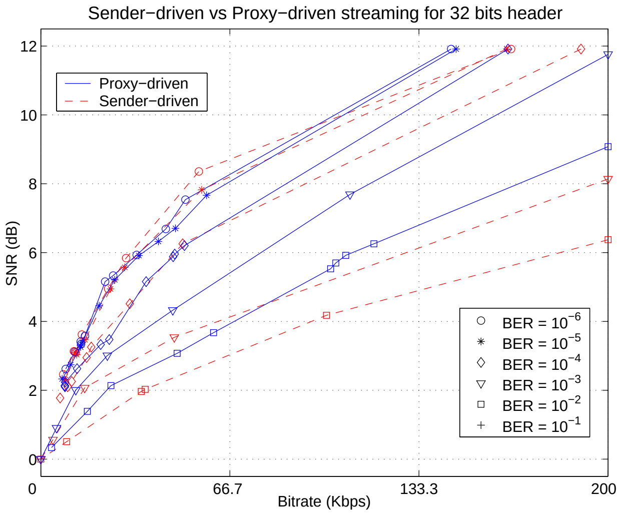

Fig. 4. RD performance of Sender-driven and Proxy-driven streaming for a wireless last hop

generator is initialized to the same seed for each of the systems under comparison. The packets have a compressed header size of 32 bits [11].

For this scenario *Sender-driven* is the system introduced in [3, 4]. It can be seen from Figure 4 that for BERs $10^{-6}, 10^{-5}$ both systems perform similarly. However, at BER $10^{-4}$ *Proxy-driven* starts performing better than *Sender-driven* for bit rates $\geq$ 50 Kbps. As the BER increases even further, the difference in performance becomes even more exaggerated and now *Proxy-driven* performs uniformly better than *Sender-driven*. The gains in performance increase with the bitrate and reach up to 4 dB. Finally, for BER $> 10^{-2}$ both systems perform poorly. This can be explained by the fact that at such high BERs every single packet is received with a corrupted header and thus is dropped by the IP layer at the client side. Hence the client never gets a chance to see a transmitted packet and exploit the benefits of the IR transmission scheme.

As can be seen from the results, *Proxy-driven* performs significantly better than *Sender-driven* in both scenarios. The improved performance is due to the fact that the proxy can always make a better estimate than the media server of the state of both the backbone network and the last hop as well as of the state of the client. This is true since the proxy is aware, contrary to the server, of the different packets going in both directions that have reached so far the edge of the backbone network.

## VI. Conclusions

A system for distortion-rate optimized streaming to clients over lossy packet networks has been presented. Rather than streaming the data directly from a media server to a client, we propose to have a mediator, a proxy server, between the media server and the client. The proxy is located at the edge of the backbone network and coordinates the streaming process. This improves the end-to-end performance and relieves the backbone network from the traffic load created by retransmissions of media packets lost in the last hop. Furthermore, by employing a distortion-rate optimization framework for packet scheduling in a hybrid receiver/sender driven scenario our system uses the available bandwidth in a most cost-effective manner. Gains of up to 2 dB and up to 4 dB for the case when the last hop to the client is respectively wireline and wireless, are observed over state of the art sender-driven systems that are also distortion-rate optimized.

## References

[1] P. A. Chou and Z. Miao. Rate-distortion optimized streaming of packetized media. *IEEE Trans. Multimedia*, 2001. Submitted.

[2] P. A. Chou and A. Sehgal. Rate-distortion optimized receiver-driven streaming over best-effort networks. In *Proc. Packet Video Workshop*, Pittsburg, PA, April 2002.

[3] J. Chakareski, P. A. Chou, and B. Aazhang. Computing rate-distortion optimized policies for streaming media to wireless clients. In *Proc. Data Compression Conference*, Snowbird, UT, April 2002. IEEE.

[4] J. Chakareski and P. A. Chou. Application layer error correction coding for rate-distortion optimized streaming to wireless clients. In *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, Orlando, FL, May 2002.

[5] A. Sehgal and P. A. Chou. Cost-distortion optimized caching of streaming media. In *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, Orlando, FL, May 2002. IEEE.

[6] G. Cheung and T. Yoshimura. Streaming agent: A network proxy for media streaming in 3g wireless networks. In *Proc. Packet Video Workshop*, Pittsburgh, USA, April 2002. IEEE.

[7] L.A. Larzon, M. Degermark, and S. Pink. UDP lite for real time multimedia applications. Technical Report HPL-IRI-1999-001, HP Labs, Bristol, United Kingdom, April 1999.

[8] N. Seshadri and C. Sundberg. List Viterbi decoding algorithms with applications. *IEEE Trans. Communications*, pages 313–323, February 1994.

[9] C. Heegard and S.B. Wicker. *Turbo Coding*. Kluwer Academic Publishers, January 1999.

[10] J. Chakareski, P. Chou, and B. Girod. Computing rate-distortion optimized policies for hybrid receiver/sender driven streaming of multimedia. In *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, November 2002. Submitted.

[11] C. Bormann et al. Robust header compression (ROHC). Technical Report RFC 3095, IETF, http://www.ietf.org/rfc/rfc3095.txt, July 2001. Proposed standard.