Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 39524, 8 pages
doi:10.1155/2007/39524

Research Article

Packet Media Streaming with Imprecise Rate Estimation

Dan Jurca and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Institute, 1015 Lausanne, Switzerland

Received 2 November 2006; Accepted 19 December 2006

Recommended by Guobin (Jacky) Shen

We address the problem of delay-constrained streaming of multimedia packets over dynamic bandwidth channels. Efficient streaming solutions generally rely on the knowledge of the channel bandwidth, in order to select the media packets to be transmitted, according to their sending time. However, the streaming server usually cannot have a perfect knowledge of the channel bandwidth, and important packets may be lost due to late arrival, if the scheduling is based on an over-estimated bandwidth. Robust media streaming techniques should take into account the mismatch between the values of the actual channel bandwidth and its estimation at the server. We address this rate prediction mismatch by media scheduling with a conservative delay, which provides a safety margin for the packet delivery, even in the presence of unpredicted bandwidth variations. We formulate an optimization problem whose goal is to obtain the optimal value for the conservative delay to be used in the scheduling process, given the network model and the actual playback delay imposed by the client. We eventually propose a simple alternative to the computation of the scheduling delay, which is effective in real-time streaming scenarios. Our streaming method proves to be robust against channel prediction errors, and performs better than other robustness mechanisms based on frame reordering strategies.

1. INTRODUCTION

Media streaming over the internet is experiencing solid growth and success in the past few years. However, the inherent best-effort characteristics of the underlying transport medium, need to be matched by intelligent streaming mechanisms in order to ensure the success of media applications. In particular, flexible rate adaptation mechanisms must be deployed to compensate for the bandwidth variability observed in the internet. When effective media transcoding capabilities are excluded due to application constraints, packet selection and scheduling represent a powerful solution for adapting the media content transmission to the available resources offered by the transport network. Under timing constraints imposed by a fixed playback delay, efficient media scheduling solutions must rapidly adapt the media packets transmission, to the available channel resources, in order to optimize the quality of service at the client.

Among the most popular scheduling schemes, a first family of methods models the underlying network in a stochastic framework. Given a permitted playback delay at the client, they attempt to maximize the expected received media quality by packet dropping or retransmission. They transform the scheduling decision into a stochastic optimization problem whose result achieves the maximum possible steaming quality for the end user [1]. However, even for simple channel models, the optimal solution requires complex algorithms that necessitate large computational resources. Hence, low delay streaming applications cannot rely on this mechanism for successful data transfer over the network. A second set of scheduling solutions rather considers the network topology and parameters as known in advance and realizes a deterministic scheduling of the packets, in order to maximize the received media quality. Such solutions are simpler, and can be employed in real-time applications. Polynomial time algorithms can take fast decisions on the pool of media packets [2, 3], in order to optimize the streaming process to the available channel conditions. However, this method may prove inaccurate and prone to errors in the case where the exact channel parameters are not fully known, or inexactly predicted by available channel estimation protocols.

Even the best rate estimation algorithms are not able to follow the rate variations of the channel, and often work on a coarser timescale [4]. Since channel prediction errors are inevitable and can lead to late arrivals of important media packets, the streaming server has to design robust packet selection and scheduling strategies against estimation mismatches. Our proposed method relies on a simple FIFO

scheduling mechanism. However, we increase the algorithm's robustness by using a conservative virtual playback delay, smaller than the playback delay imposed by the client [5]. The scheduling process considers the conservative playback delay as the hard deadline for packet arrival at the client, hence it is more aggressive in the packet selection process. On the other side, the difference between the conservative scheduling delay and the effective playback delay after which the client starts playing the video, transparently compensates for the eventual late packet arrivals due to the erroneously estimated end-to-end channel rate variations.

Overall, we observe that a very conservative scheduling delay tends to limit the selection of transmitted media data to only a few packets, which penalizes the quality at the receiver. Alternatively, a scheduling delay that is too close to the effective playback delay may result in late arrival of packets, which also penalizes the quality. Hence, the purpose of this work is to analyze the trade-off between robustness against channel prediction errors, and packet selection limitations, observed as a result of tighter scheduling constraints.

The rest of this paper is organized as follows: Section 2 presents an overview of existing work in the domain of robust media streaming. We formulate in Section 3 an optimization problem whose goal is to find the optimal conservative delay used in the scheduling process, which maximizes the quality of the received video for a given channel rate model, and a given playback delay at the client. We discuss the complexity of the exact solution for the optimization problem and we present a fast solution in Section 4. Section 5 presents our simulation results and Section 6 concludes this paper.

2. RELATED WORK

Robustness to network failures is one of the necessary attributes of a successful media application. Application layer tools have been designed for providing the required flexibility in order to cope with network variations. The authors of [6, 7] present an overview of video coding techniques that confer flexibility and robustness to the streaming process. Error resilient video encoding and error-concealment strategies at the client side are detailed. These techniques can be further enhanced with network layer error-robustness strategies like ARQ or FEC [8, 9]. While these mechanisms offer the flexibility needed in order to cope with network channel errors and variations, their design is based on the knowledge of network parameters. Their functionality depends on the accuracy of the channel estimation, hence when these estimations are inexact, they are susceptible to failure. Intelligent scheduling on a packet level can adapt the media streaming decisions in case of network parameter variability, and add an extra layer of flexibility in the wake of adverse network conditions (e.g., bandwidth shortage, or variable transmission delays and jitter).

Informed scheduling decisions optimize the received media quality under network resources constraints. In the Rate-Distortion framework presented in [1], the scheduling algorithm takes an optimal decision (transmission policy) for each media packet/set of packets, based on the parameters of the channel model. The channel is stochastically modelled, and the optimal scheduling solution comes at the expense of complex computations and large delays [10, 11]. On the other hand, deterministic scheduling algorithms [2, 12] are faster, but require exact knowledge of channel parameters, and are prone to errors in case of unpredicted network variability. Previous works [13, 14] enhance the robustness to channel prediction errors, by designing a new scheduling model, in which the packets/frames in a bitstream are rearranged. The most important parts of the bitstream are advanced ahead of the less important ones, so that they are scheduled for transmission with higher priority. Such mechanisms increase the probability of successful transmission of information necessary for correct decoding, even in adverse network scenarios. While these methods increase the robustness to network delay fluctuations, they are also more demanding in terms of codec buffer sizes and computation.

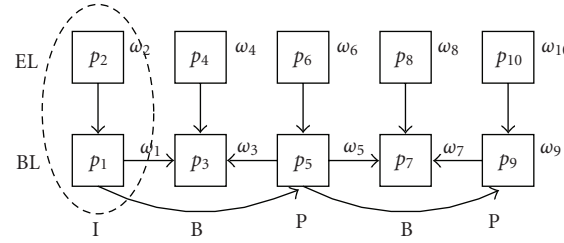

Figure 1: Video packets as a directed acyclic graph.

In this work, we rather enforce a FIFO scheduling mechanism because of its simplicity and efficient use of buffering resources. However, we propose to increase its robustness to channel estimation errors by scheduling packets with a virtual playback delay, smaller than the playback delay imposed by the client. The difference between the scheduling delay and the playback delay after which the client starts playing the video, can transparently absorb the effects of the erroneously predicted end-to-end rate variations on packet arrival times.

3. STREAMING WITH CONSERVATIVE DELAY

3.1. System overview

We consider a single path streaming scenario between a server S and a client C. The media stream can either be prestored at the server (VoD), or can be obtained in real time (real-time streaming). The video content is encoded into one or more layers and fragmented into network packets such that one packet contains information related to one frame and one video layer. Let $P = \{p_1, \ldots, p_n\}$ be the set of available packets at the server, with n representing the total number of packets. Similarly to [3], each packet p_i is completely characterized by its size s_i, its decoding deadline t_i, its importance ω_i, and its list of dependency packets A_i, which are necessary for a correct decoding (see Figure 1).

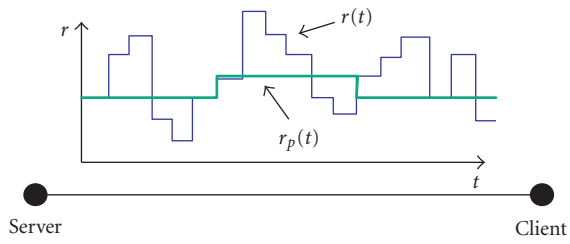

Figure 2: Network end-to-end model with rate variations $r(t)$ and estimated rate $r_p(t)$.

Table 1: Example parameter values for conservative delay scheduling.

Instantaneous rate (kbps)	420
Predicted rate (kbps)	450
Packet size s_i (bits)	8000
Packet weight ω_i	1000
Decoding deadline t_i	0
Playback delay Δ (ms)	200
Conservative playback delay δ (ms)	180
Time t (ms)	0

The intermediate network between S and C is modelled as an end-to-end channel characterized by the variable rate $r(t)$. While we consider no link error in our model, packets can still be lost from a media application perspective, due to late arrivals. The server S estimates on a periodic interval, the available channel rate $r_p(t)$, using any estimation mechanism Γ (see Figure 2). Based on that estimation, the streaming application employs a generic scheduling algorithm Ψ that decides the subset of packets $\pi \subseteq P$ that are sent in a FIFO order to the client, so that the reconstructed video quality is maximized, given the playback delay Δ imposed by the client. The video quality measure Ω can be computed at the client as

$$\Omega = \Omega_S(\pi) - \Omega_L(\pi), \tag{1}$$

where $\Omega_S(\pi) = \sum_i \omega_i$, for all $p_i \in \pi$ represents the quality of the video packets selected for transmission, and $\Omega_L(\pi) = \sum_i(\omega_i \cdot P_i)$ represents the video quality degradation due to packets that cannot be decoded because of late arrivals at the client. P_i represents the probability that packet p_i arrives past its decoding deadline at the client. These late arrivals are caused by channel bandwidth variations, and inaccuracy in the rate estimation used by the server. Indeed, the estimation of the available rate in the future time instants is generally not perfect, and often not able to exactly follow the frequent variations of the bandwidth.

We propose to modify the scheduling strategy, in order to be robust to over estimations of the channel rate. We define a virtual playback delay, or scheduling delay δ, which is used by the server to compute the subset of packets to be sent. As δ is smaller than the actual playback delay Δ, the server will select a reduced number of packets for transmission (Ω_S decreases), but the selected packets have a lower probability to be lost (Ω_L increases). In other words, π now contains only packets that can reach the client before their decoding deadline ($t_i + \delta$) with a streaming rate r_p, and each packet p_i is scheduled and transmitted only once. The choice of the virtual playback delay becomes obviously a trade-off between source quality and robustness to rate variations, and its optimization is proposed in the next sections.

3.2. Illustrative example

We demonstrate the rationale behind our proposed mechanism by a concrete example. Imagine that server S needs to decide at time t whether to send packet p_i to the client C or not. The scheduling decision is based on the predicted network rate at moment t, $r_p(t)$, the size s_i, weight ω_i, dependency list A_i and decoding deadline t_i of packet p_i, and on the conservative playback delay δ. In the same time, C expects packet p_i before time $t_i + \Delta$, so that it can successfully decode it.

For the sake of clarity, assume that the list $A_i = \emptyset$, for example, packet p_i can be independently decoded at C, and that the server's buffer does not contain any other media packets except p_i. The rest of the parameters are set according to Table 1.

Observe that S will take the decision to send the packet on the network after computing the expected arrival time at the client: $T_p = t + s_i/r_p(t) \approx 177\,\text{ms} \leq 180\,\text{ms} = t_i + \delta$. Even if the channel rate is overestimated and packet p_i arrives at the client at $T_a = t + s_i/r(t) \approx 190\,\text{ms} > t_i + \delta$, packet p_i will still arrive on time for successful decoding at the client, as $t_i + \Delta = 200\,\text{ms}$.

On the contrary, imagine the same procedure is applied to packet p_j, under the same conditions, except $s_j = 9.000$ bits and the scheduler does not use the conservative delay δ, rather directly the playback delay Δ. S decides to send the packet as $T_p = t + s_j/r_p(t) = 200\,\text{ms} \leq 200\,\text{ms} = t_i + \Delta$. However, packet p_j is useless for the client as it arrives past its decoding deadline: $T_a = t + s_j/r(t) \approx 220\,\text{ms} > t_i + \Delta$. In such a case packet p_j consumes network resources that could be used more effectively.

Finally, please observe that in the case where S uses the conservative delay δ in scheduling packet p_j, the decision would be to drop the packet, as it would arrive late. This insight lies the ground for the trade-off between robustness against channel prediction errors, and packet selection limitations, observed as a result of tighter scheduling constraints.

3.3. Optimization problem

The virtual playback delay δ used by the scheduler represents a compromise between a conservative selection of packets that minimizes the probability of late arrivals, and the selection of a sufficient number of packets for an effective quality. Given the video sequence, the quality metric Ω, the scheduling strategy Ψ, the rate estimation algorithm Γ, and the playback delay Δ, the optimization problem translates into finding the optimal conservative delay $\delta \leq \Delta$ to be used by

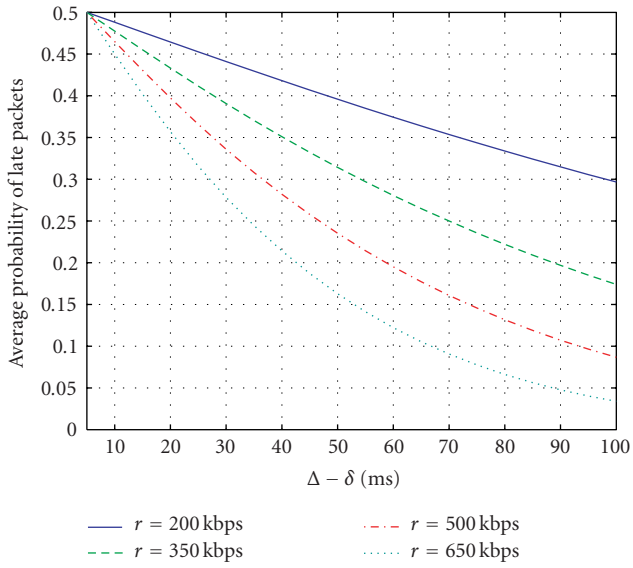

Figure 3: Average probability of late packets ($\Delta = 300$ ms).

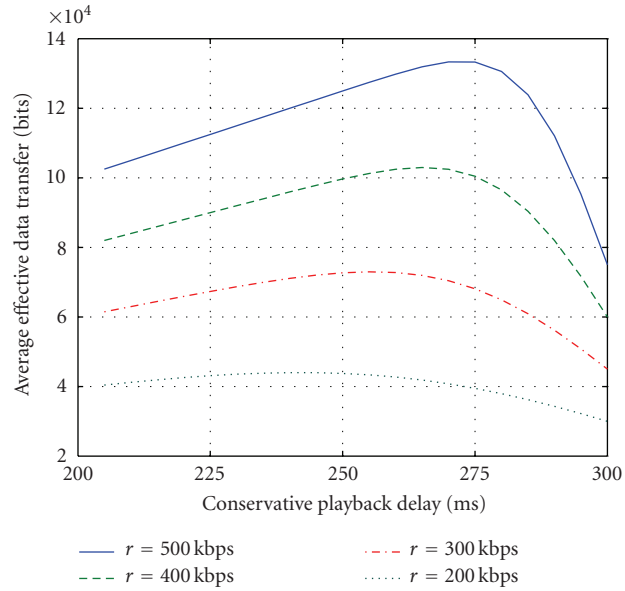

Figure 4: Effective average data transfer ($\Delta = 300$ ms).

the streaming application, in order to maximize the received video quality Ω, for a given channel model

$$\delta^* = \arg\max_{\forall \delta \leq \Delta} \Omega(\delta). \tag{2}$$

In general, this optimization problem does unfortunately not provide any simple solution. Even for fixed Ψ, Γ, and Δ, the scheduling policy π is not constant with the choice of δ, hence finding the optimal solution for the problem has combinatorial complexity. However, for small values of Δ (as in practical real time streaming scenarios), δ^* can be accurately approximated in real time. In the next section, we present our approach towards finding an appropriate solution, based on heuristics from real-time video streaming.

4. FINDING THE CONSERVATIVE DELAY

4.1. General solution

On the one hand, the quality measure $\Omega_L(\pi)$ depends only on the difference $\Delta - \delta$, for a given transmission policy π and the channel model. Very conservative values for δ will ensure a big difference $\Delta - \delta$, hence more margin in dealing with rate prediction errors, and consequently a smaller value for Ω_L (see Figure 3).

On the other hand, the quality measure $\Omega_S(\pi)$ depends only on the scheduled packets according to the predicted rate $r_p(t)$ and δ. Interestingly, our experiments show that, for a given channel model, Ω_S does not vary much with δ, as long as δ is large enough to accommodate the transmission of the largest video packets of the sequence.

Let $R^i(\Delta)$ be the cumulative rate of the channel up to time $t_i + \Delta$: $R^i(\Delta) = \int_0^{t_i+\Delta} r dt$, and $R_p^i(\delta)$ be the cumulative estimated rate up to time $t_i + \delta$: $R_p^i(\delta) = \int_0^{t_i+\delta} r_p dt$. For given δ and Δ, we define the effective data transfer $\mathcal{C}_\Delta^\delta(i)$ on the time interval $[0, t_i+\Delta]$, as the amount of data scheduled according to r_p before $t_i + \delta$, and received before $t_i + \Delta$ according to r:

$$\mathcal{C}_\Delta^\delta(i) = R_p^i(\delta) \cdot Pr\{R_p^i(\delta) \leq R^i(\Delta)\}. \tag{3}$$

An illustration of the effective data rate transfer is given in Figure 4.

Given this measure, we transform the original optimization problem into a new one that chooses δ in order to maximize $\mathcal{C}$, defined as

$$\delta^* = \arg\max_{0 \leq \delta \leq \Delta} \mathcal{C}_\Delta^\delta(i). \tag{4}$$

$\mathcal{C}_\Delta^\delta(i)$ is invariant in time, as long as the channel model does not change, hence it can be computed at any t_i. The previous optimization problem translates into maximizing the chances of every packet p_i, scheduled for transmission at time t, to reach its destination by time $t + \Delta$. Unlike the original optimization problem of (2), (4) depends only on the channel model, hence it is easy to solve, once this model is known. It can be noted that both optimization problems are equivalent in the case of a smooth video model (the video packets have the same size and importance, and there are no dependency among them). We later show in Section 5 that even in realistic video streaming scenarios the solution obtained for this problem is a very good approximation of the optimal solution.

4.2. Example channel model

We now develop all necessary relations for a typical channel modelled as a discrete-time system, with a sampling interval of T_s seconds. The network can communicate a maximum of $r_i T_s$ bits of data in the time interval $[iT_s, (i+1)T_s]$, where r_i is

the available bandwidth of the channel in the ith time interval. The channel rate r_i is given as a Gaussian autoregressive process of the form

$$r_i = \mu + (1-\alpha)\sum_{j=0}^{\infty}\alpha^j n_{i-j}, \quad j \in \mathbf{Z},\ n_k = 0,\ \forall k < 0. \quad (5)$$

Each n_j is an independent zero mean Gaussian random variable with variance σ^2, α is a modelling parameter, and μ denotes the average available bandwidth. The validity of that model for internet traffic traces on time scales of milliseconds up to a few seconds has been verified in [15].

A simple auto-regressive prediction model is used for bandwidth estimation at the server, where the available rate of the network in the next time interval, $k+1$, is given by

$$r_{k+1} = \gamma\frac{\sum_{j=1}^{k-1} r_j}{k-1} + (1-\gamma)r_k, \quad (6)$$

where γ is the prediction coefficient. The estimation is run periodically, on time windows of size T_p. While instantaneous rate variations of the channel can happen on very small time scales (of tens to hundreds of milliseconds), the fastest estimation mechanisms provide accurate results on time intervals of the size of a few round-trip times (e.g., one second or more), and prediction inaccuracies cannot be avoided.

Assuming that $t_i + \Delta = k \cdot T_s \le T_p$, with k an integer,[1] we can compute

$$R^i(\Delta) = k\cdot\mu + \sum_{j=0}^{k}(1-\gamma)\cdot\gamma^{j-1}\cdot\sum_{l=1}^{k-j} n_l. \quad (7)$$

Finally, $\mathcal{S}_i$ denotes the cumulative size of the transmitted packets up to packet p_i : $\mathcal{S}_i = \sum_{j=1}^{i} s_j$, for all $p_j \in \pi$. The probability that a packet arrives too late at the receiver, P_i, can be computed as

$$P_i = Pr\left\{\mathcal{S}_i > R^i/\mathcal{S}_i \le R_p^i\right\}. \quad (8)$$

Since R^i is a normal random variable and R_p^i is a known constant, given any δ and Δ, the error probabilities P_i can be easily computed with the help of the $erfc$ function.

4.3. Scheduling algorithm

While the presented robustness mechanism is generic, and can be applied to any packet scheduling algorithm, in this section we describe the specific algorithm employed in the experimental phase of this paper.

The algorithm is an adaptation of the LBA scheduling algorithm introduced in our prior work [12], to the single-path network scenario presented above. In short, the algorithm performs a greedy scheduling of the most valuable packets first. Less valuable packets are scheduled only if the network capacity permits, and only if they do not lead to the loss of a more valuable packet already scheduled (due to subsequent late arrivals at the client).

First, the n network packets are arranged in descending order of their weight, obtaining a new representation of the encoded bitstream, $P' = \{p'_1, p'_2, \ldots, p'_n\}$. Then, the algorithm attempts a greedy scheduling of the packets on the network link, starting with the most important one. To decide which action to take on each packet p'_i, the algorithm first attempts to schedule all ancestors that have not been scheduled yet. If one of them cannot be scheduled, then the algorithm automatically drops the packet p'_i. This ensures that our algorithm does not waste network resources on transmitting network packets that cannot be correctly decoded at the receiver.

Finally, all packets marked to be transmitted, are reordered according to their decoding deadlines before transmission. When a new packet is inserted for transmission, it triggers a new packet ordering. If packet p'_i can be inserted, without compromising the arrival time of any other already scheduled packet, then it is marked for transmission. Otherwise, packet p'_i is dropped. Please observe that the scheduling algorithm can be run on the total video sequence to be streamed, in the case of VoD streaming, or on a limited window of video packets in the case of real-time streaming.

The total complexity of the scheduling algorithm is driven mainly by the sorting and insertion operations. While the sorting can be performed by any algorithm in time $O(n\log n)$, the insertion of each packet p'_i requires a complete parse through all previously scheduled packets. Hence the total complexity of the algorithm is $O(n^2)$.

5. SIMULATIONS

We discuss the performance of the streaming application with conservative delay and we compare the results obtained by our heuristic solution for δ with the optimal one, and with other frame reordering techniques. We scalably encode the *foreman_cif* sequence (130 frames) using MPEG4-FGS, at 30 frames per second, with a GOP structure of 31 frames (IPBPBPB...). By splitting the bitplanes, we encode one BL and 2 ELs of average rates of 260 kbps. In all our experiments we use the simple packet scheduling algorithm as presented above. We set the weights ω_i of the packets as a function of their relative importance to the encoded bitstream (depending on the type of encoded frame, I, P, or B, and on the encoded layer they represent, BL, EL1, or EL2), as illustrated in Figure 1. In a first approximation, we choose the following packets weights: 5 for I frame BL packets, 4 for the P frame BL packets, 3 for the B frame BL packets, 2 for the EL1 packets, and 1 for the EL2 packets [3].

For the channel model and estimation mechanism, we set the required parameters to $\alpha = \gamma = 0.8$, $T_s = 20$ ms, $T_p = 1$ s, and we vary $\sigma^2 \in [100, 250]$, according to the channel average rate. These values insure realistic channel variations on small time scales around the average bandwidth value. Finally, we set $\Delta = 200$ ms.

[1] The extension of the computation for the general case, on multiple prediction intervals, and when k is not an integer, is straightforward, and omitted in this manuscript.

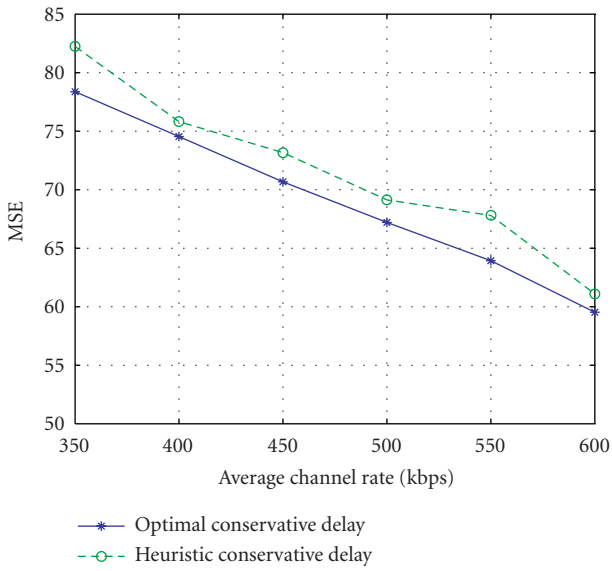

Figure 5: Quality evaluation for scheduling with heuristic and optimal δ.

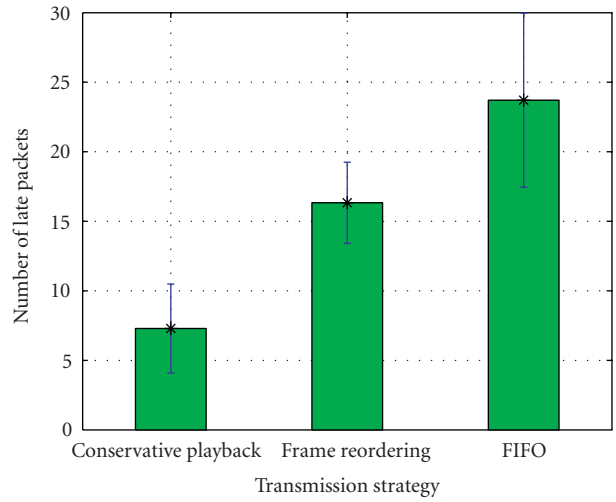

Figure 6: Late packets: conservative δ; frame reordering; FIFO scheduling.

Table 2: δ^* and δ for various average channel rates.

Rate (kbps)	350	400	450	500	550	600
Optimal δ^* (ms)	163	156	172.5	161	154	155.5
Heuristic δ (ms)	172	170	168	167	166	165
$\frac{\Omega(\delta^*)-\Omega(\delta)}{\Omega(\delta^*)}$ (%)	4.94	1.71	3.53	2.86	6.04	2.63

First we compare the results obtained by streaming with the heuristic δ, computed according to (4), and the optimal δ^*, obtained after a full search through all possible values for $\delta \in [0, \Delta]$. We use different channel average rates and we average over 10 simulations for each case. The results are presented in Figure 5. We observe that for all simulated rates, our results in terms of MSE are very close to the optimal ones. This validates our simplification to the original optimization problem, presented in Section 4. In the same time, Table 2 presents the obtained values for the heuristic and optimal δ for the same channel conditions as above, along with the relative error between the streaming performances. We observe that the values are very close and that δ^* is in general more conservative than δ. An explanation to this phenomenon resides in the fact that the sequence under consideration does not present any scene changes and the packet sizes remain constant in time.

Next, we compare the proposed conservative δ streaming with other frame reordering streaming techniques. We use a simple technique similar to the one presented in [13], which brings forward all I and P frames by two positions in the original bitstream before scheduling. Both techniques are compared in terms of number of late packet arrivals with a simple FIFO scheduling scheme that is unaware of channel rate variations. Simulation results are averaged over 100 channel realizations for an average rate of 500 kbps. Figure 6 presents the number of late packets for each of the 3 schemes with the 95% confidence intervals. We observe that the conservative δ scheme performs the best in terms of average number of late arrivals, due to the fact that the application can transparently use the difference $\Delta - \delta$ to compensate for unpredicted channel rate variations. Figure 7 presents one scheduling example for the conservative δ and frame reordering techniques. We observe that in the case of frame reordering, the strategy trades off a higher confidence in receiving I and P frames on time, at the expense of less important B frames. Hence, some B frames are lost due to late arrivals. On the contrary, the conservative δ strategy manages to schedule a similar amount of packets, and uses the extra time $\Delta - \delta$ to minimize the impact of rate variations on late arrivals. Hence, less packets are late at the receiving end of the application.

Finally, we test the proposed conservative delay scheduling method on network rate traces generated with the help of the ns-2 simulator in the presence of background traffic. We simulate 10 background flows that use the same bottleneck link as our media stream. These flows are generated according to the on/off exponential distribution, with average rates between 100 and 300 kbps. The available instantaneous rate for our streaming application is considered to be the difference between the total link bandwidth and the aggregated instantaneous rate of the background traffic. Even if the average available rate stays constant, instantaneous rate variations can be larger than 100%. We compare the performance of the scheduling obtained by using the heuristic and the optimal conservative delays, respectively, by averaging the obtained results over 10 randomly generated network rate traces. Results are presented in Figure 8 for average network rates of 300 and 450 kbps. We observe that the results are very close, even if the exact channel model is not known when the conservative delay is computed, and the channel estimation

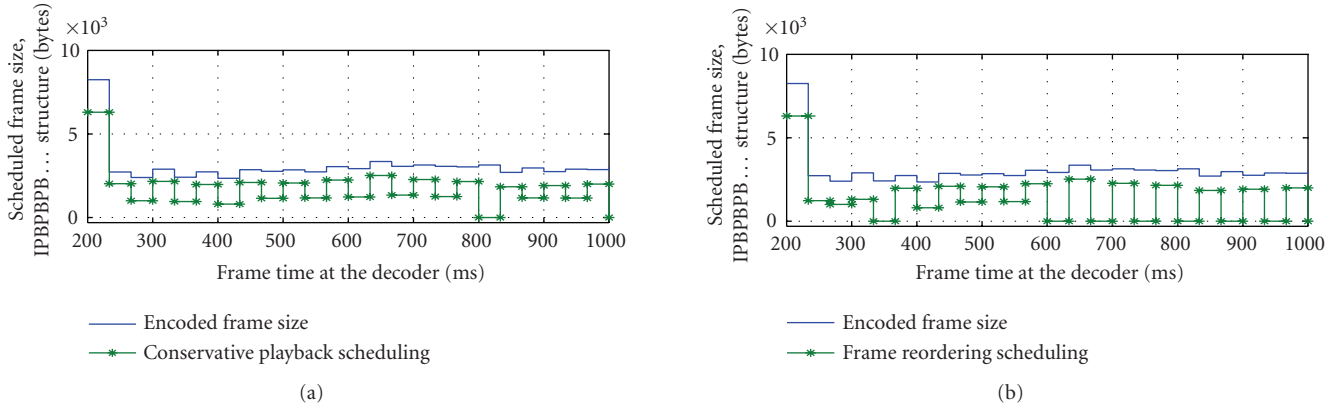

Figure 7: Example of conservative δ and frame reordering scheduling.

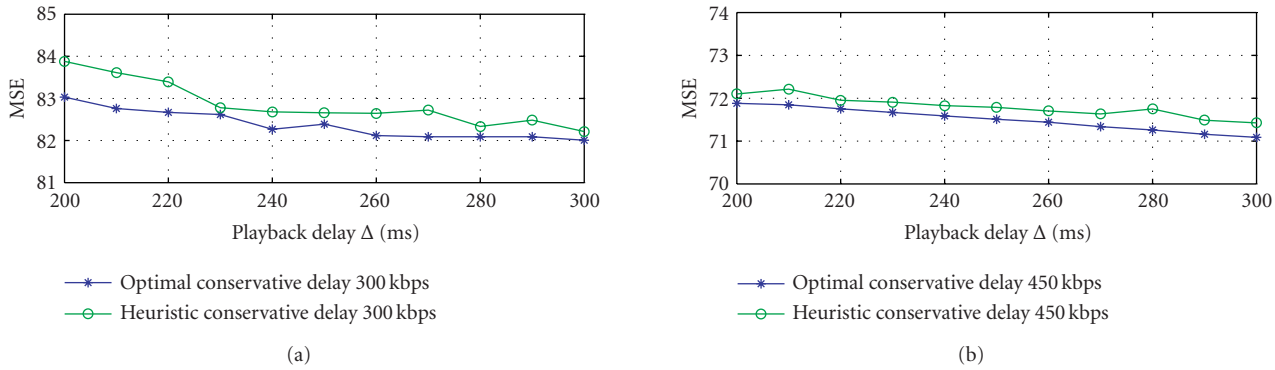

Figure 8: Quality evaluation for scheduling with heuristic and optimal δ for ns-2 network rate traces.

method is imperfect.[2] Results show that being conservative in terms of scheduling delay and initial channel rate estimate, increases the robustness of the streaming application, without significantly penalizing the received video quality. It indicates that our method is robust even in extreme cases when exact information related to the channel model is not available.

6. CONCLUSIONS

We present a new mechanism to improve the robustness of adaptive media stream scheduling algorithms against network channel variability and estimation inaccuracies. By using a conservative virtual playback delay in the scheduling process, we compensate for possible prediction errors. The difference between the conservative and actual playback delay imposed by the client transparently absorbs the negative effects of inexact rate estimation (e.g., increased packet delay at the client due to channel variations). We propose a method to determine the value of the conservative delay, as a trade-off between source quality, and robustness to bandwidth variations. The proposed solution is generic and can be employed with any given streaming mechanism. Results show that being conservative in choosing the scheduling delay pays off, even if the exact channel model is unknown (e.g., on simulated network rate traces with competing background traffic) and the rate estimation mechanism only approximates the channel rate variations over time. The simplicity of our solution and its effectiveness make it appropriate for any real-time streaming mechanism over best-effort networks.

[2] For more details on efficient bandwidth estimation mechanisms, we refer the reader to [4].

ACKNOWLEDGMENT

This work has been supported by the Swiss National Science Foundation under Grant no. PP-002-68737.

REFERENCES

[1] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 390–404, 2006.

[2] K. Chebrolu and R. R. Rao, "Bandwidth aggregation for real-time applications in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 388–403, 2006.

[3] D. Jurca and P. Frossard, "Distortion optimized multipath video streaming," in *Proceedings of the International Packet Video Workshop*, Irvine, Calif, USA, December 2004.

[4] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: efficient available bandwidth estimation for network paths," in *Proceedings of Passive and Active Measurement Workshop (PAM '03)*, La Jolla, Calif, USA, April 2003.

[5] D. Jurca and P. Frossard, "Media streaming with conservative delay on variable rate channels," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '06)*, pp. 1841–1844, Toronto, Ontario, Canada, July 2006.

[6] B. Girod and N. Färber, "Feedback-based error control for mobile video transmission," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1707–1723, 1999.

[7] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, 2000.

[8] F. Wu, H. Sun, G. Shen, et al., "SMART: an efficient, scalable, and robust streaming video system," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 2, pp. 192–206, 2004.

[9] D. G. Sachs, I. Kozinetsev, M. Yeung, and D. L. Jones, "Hybrid ARQ for robust video streaming over wireless LANs," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '01)*, pp. 317–321, Las Vegas, Nev, USA, April 2001.

[10] R. Zhang, S. L. Regunathan, and K. Rose, "End-to-end distortion estimation for RD-based robust delivery of pre-compressed video," in *Proceedings of the 35th IEEE Annual Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 210–214, Pacific Grove, Calif, USA, November 2001.

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

[12] D. Jurca and P. Frossard, "Video packet selection and scheduling for multipath streaming," to appear in *IEEE Transactions on Multimedia*.

[13] S. Wee, W.-T. Tan, J. Apostolopoulos, and M. Etoh, "Optimized video streaming for networks with varying delay," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME '02)*, vol. 2, pp. 89–92, Lausanne, Switzerland, August 2002.

[14] J.-W. Ding, Y.-M. Huang, and C.-C. Chu, "An end-to-end delivery scheme for robust video streaming," in *Proceedings of 2nd IEEE Pacific Rim Conference on Multimedia (PCM '01)*, vol. 2195 of *Lecture Notes In Computer Science*, pp. 375–382, Bejing, China, October 2001.

[15] A. Sang and S.-Q. Li, "A predictability analysis of network traffic," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFCOM '00)*, vol. 1, pp. 342–351, Tel Aviv, Israel, March 2000.