SCHOOL OF ENGINEERING - STI
SIGNAL PROCESSING INSTITUTE
Dan Jurca and Pascal Frossard

CH-1015 LAUSANNE

Telephone: +4121 6936874
Telefax: +4121 6937600
e-mail: dan.jurca@epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PACKET SELECTION AND SCHEDULING FOR MULTIPATH VIDEO STREAMING

Dan Jurca and Pascal Frossard

Swiss Federal Institute of Technology Lausanne (EPFL)

Signal Processing Institute Technical Report

TR-ITS-2004.025

November 9th, 2004

Part of this work has been submitted to IEEE TMM

This work has been supported by the Swiss National Science Foundation.

Packet Selection and Scheduling for Multipath Video Streaming

Dan Jurca and Pascal Frossard
Signal Processing Institute
EPFL, Lausanne - Switzerland
Email: {dan.jurca, pascal.frossard}epfl.ch

***Abstract*— This paper addresses the problem of choosing the best streaming policy for distortion optimal multipath video delivery, under delay constraints. The streaming policy consists in a joint selection of the video packets to be transmitted, as well as their sending time, and the transmission path. A simple streaming model is introduced, which takes into account the video packet importance, and the dependencies among packets, and allows to compute the quality perceived by the receiver, as a function of the streaming policy. We derive an optimization problem based on the video abstraction model, under the assumption that the server knows, or can predict the state of the network. A detailed analysis of the timing constraints in multipath video streaming provides helpful insights that lead to an efficient algorithm to solve the NP-hard streaming policy optimization problem. We eventually propose a fast heuristic-based algorithm, that still provides close to optimal performance. Thanks to its limited complexity, this novel algorithm is finally demonstrated in live streaming scenarios, where it only induces a negligible distortion penalty compared to an optimal strategy. Simulation results finally show that the proposed scheduling solutions perform better than common scheduling algorithms, and represent very efficient multipath streaming strategies for both stored and live video services.**

I. Introduction

Despite the development of novel network infrastructures, and constantly increasing bandwidth, Internet media streaming applications still suffer from limited and highly varying bandwidth, and often from packet loss. Multipath Video Streaming has recently been proposed as a solution to overcome packet network limitations. It allows to increase the streaming bandwidth by balancing the load over multiple (disjoint) network paths between the media server and the clients. It also provides means to limit packet loss effects when combined with error resilient streaming strategies [1]. The efficiency of multipath video streaming is however tied to the packet transmission strategies, that aim at offering an optimal quality of service in delay-constrained video applications.

This work addresses the problem of video packet streaming in multipath network scenario, under delay and buffer constraints. It aims at efficiently distributing the video information on the available network paths, while judiciously trading off playback delay and distortion at the receiver. This paper considers the selection of inter-dependent video packets to be transmitted (or equivalently the adaptive coding of the video sequence), and the scheduling on the different network paths, in order to minimize the distortion experienced by the end-user. The complex distortion optimization problem is a priori NP-complete, and no method can solve it in polynomial time [2]. We however propose fast solutions, that perform very close to optimal and yield to efficient real time streaming solutions.

Assuming a simple streaming model, that factors in the unequal importance of video packets, and their dependencies, we propose a detailed analysis of timing constraints imposed by delay sensitive streaming applications. This analysis allows us to identify sets of valid transmission policies, that compete for the distortion optimized multipath streaming solution. The optimal solution is computed based on a modified branch and bound algorithm [3], that applies search and pruning methods specific to the multipath streaming problem. We then propose a heuristic-based approach to the optimization problem, that leads to a polynomial time algorithm, based on load-balancing techniques. The scheduling algorithm is finally adapted to the case of real time streaming, with the help of sliding window mechanisms. Simulation results demonstrate close to optimal performances of the fast scheduling solution, for a large variety of network scenarios. Interestingly enough, the performance of the real time scheduling algorithm stays quite consistent, even for small video prefetch windows. This shows the validity of our algorithm in multipath live streaming systems, even with simple bandwidth prediction methods.

The main contributions of this paper are threefold. First, we provide a new framework for studying video packet scheduling in multipath streaming, taking into account possible buffer constraints in intermediate network nodes, on each path. Second, we derive a distortion optimization problem which takes into account the non-stationary nature of video sequences, the packet dependencies introduced by the encoding algorithm and the network status. Last, we propose a novel, fast algorithm, which solves the optimization problem, and is eventually adapted to real time streaming scenarios without significant loss in performance.

This paper is organized as follows: Section II presents the multipath streaming model and introduces the notations, that lead to a distortion optimization problem. The packet scheduling is analyzed in Section III. Based on this timing analysis, we propose optimal and heuristic algorithms to solve the distortion optimization problem in Section IV. Simulation results are presented in Section V. A description of the related work in multipath streaming is proposed in Section VI and Section VII concludes the paper.

II. Multipath Video Streaming

A. General Framework

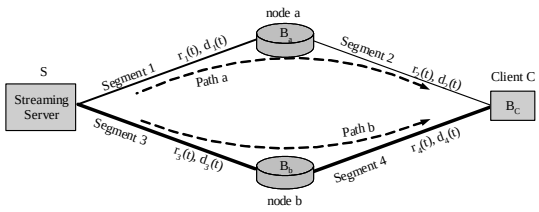


Fig. 1. Multipath Streaming Scenario. The client accesses the streaming server simultaneously through two different paths, each one composed of two segments with intermediate buffers.

We consider the simple multipath network topology represented in Figure 1. The client *C* requests a media stream from a streaming server *S*, that transmits the requested bitstream via two disjoint paths. Each network path consists in two segments connected through an intermediate node that simply forwards, after a possible buffering delay, incoming packets from the first segment, towards the client on the second segment. The intermediate nodes, simply called nodes in the remaining of the paper, represent network streaming proxies, or edge servers for example. The streaming server is connected to the channels through buffer interfaces, that can be modelled as FIFO queues. Thus, the channels drain the packets from the buffers, in the same order in which the server places them into the buffers. The network channels between the server and the client can be represented as variable bandwidth, lossless links. The variable nature of the bandwidth implies that the rate at which the channels drain data placed in the server's buffers, changes as a function of time. At the other end, the client waits for an initial playback delay Δ after its request for a stream. It then starts decoding the media stream, and plays it continuously.

During the streaming session, the server selects a subset of the pre-encoded media packets to communicate to the client, taking into account the available bandwidth on the different network paths, and buffer fullness in the nodes, or at the receiver. The work presented in this paper addresses the selection of the packets that should be communicated to the client, as well as the network path they need to follow. It has to be noted that the topology could present several disjoint paths, and several nodes on each path. Without loss of generality however, we consider in this paper only the two-path scenario presented in Figure 1, for the sake of clarity.

B. Streaming Model and Notations

In the multipath streaming topology represented in Figure 1, each network segment i is characterized by an instantaneous rate $r_i(t)$ and an instantaneous latency $d_i(t)$. The rate $r_i(t)$ is the total bandwidth allocated to the streaming application on segment i at time instant t. Equivalently, we denote the cumulative rate on segment i, up to time instant t, by $R_i(t) = \int_0^t r_i(u)du$. Additionally, we assume that no packet is lost on the network segments, except those induced by late arrivals or buffer overflows, and that the order of the packets is not changed between two successive nodes. The intermediate nodes $\{a, b\}$ have buffers of capacity B_a and respectively B_b, that are available to the streaming session. The client has a playback buffer of capacity B_c.

The video sequence is encoded into a bitstream using a scalable (layered) video encoder. The bitstream is then fragmented into network packets under the general rule stating (i) that each network packet contains data relative to at most one video frame, and (ii) that an encoded video frame can be fragmented into several network packets. Let $P = \{p_1, p_2, ..., p_N\}$ be the chronologically ordered sequence of N network packets, after fragmentation of the encoded bitstream. Each network packet p_n is characterized by its size s_n in bytes, and its decoding timestamp t_n^d. From the client viewpoint, all the video packets are not equivalently valuable, due to the non-stationary nature of the video information. Therefore, each network packet can be characterized by a weight ω_n, that represents the reduction in the distortion perceived by the client, in the case where packet p_n is successfully decoded[1].

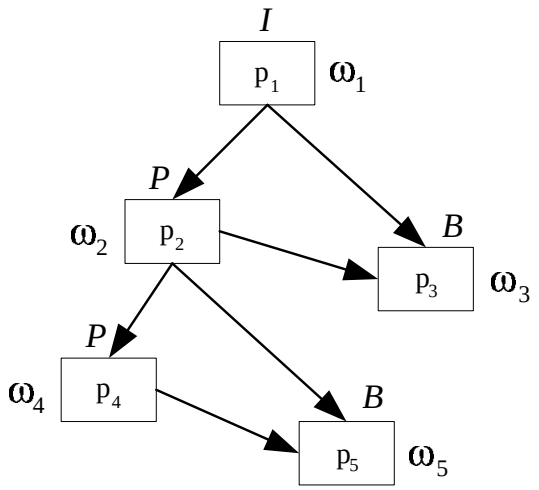


Fig. 2. Directed acyclic dependency graph representation for a typical MPEG encoded video sequence (one network packet per frame, with IPBPB format).

Additionally, in most video encoding schemes, packets have generally dependencies between them. In other words, the successful decoding of one packet p_n is contingent on the successful decoding of some other packets, called *ancestors* of p_n. The successful decoding of one packet may depend on the correct decoding of several ancestors, and we denote by A_n, the set of ancestors of packet p_n. Such dependencies can be represented by a directed acyclic dependency graph [4], as shown in Figure 2. The nodes in the graph represent the network packets and are characterized by their individual weights, and directed edges represent dependencies between packets and their ancestors.

We denote by $\pi = (\pi_1, \pi_2, ..., \pi_N)$ the transmission policy adopted by the streaming server. The policy π_n used for packet p_n consists in a couple a variables $[q_n, t_n^s]$ that respectively represent the path chosen for packet p_n, and its sending time.

[1]We refer to a successfully decoded packet as a network packet that is received and correctly decoded by the client before it's decoding time.

It completely characterizes the server behavior with respect to packet p_n under the general policy vector π. In the multipath network scenario presented hereabove, the server can decide to send packet p_n on paths *a* or *b*, or simply to drop the packet without sending it. Therefore, the action imposed on packet p_n can be written as:

$$q_n = \begin{cases} a & \text{if packet } p_n \text{ is sent on path } a \\ b & \text{if packet } p_n \text{ is sent on path } b \\ 0 & \text{if packet } p_n \text{ is dropped.} \end{cases}$$

Let Π be the set of all the feasible policies π, in the network scenario under consideration in this paper. Remember that packets are sent sequentially on a path, and that the streaming strategy aims at avoiding buffer overflows that result in packet loss.

Finally, in our streaming model, a packet is decoded by the receiver only if its arrival time, t_n^c, is smaller than its decoding deadline, i.e., if $t_n^c \leq t_n^d + \Delta$ where t_n^d represents the decoding timestamp of packet p_n. We assume here, without loss of generality, that the client request has been sent at time $t = 0$, and that the decoding timestamp of the first packet p_1 is set to 0. The decoding time at the receiver is further neglected. Under these assumptions, and taking into account packet dependencies, the successful decoding of a packet p_n under the streaming strategy $\pi \in \Pi$, can be represented by the binary variable $\varphi_n(\pi)$, where $\varphi_n(\pi) = 1$ if the packet arrives on time at the decoder, and if all its ancestors have been successfully decoded. In other words, we can write:

$$\varphi_n(\pi) = \begin{cases} 1 & \text{if} \quad \begin{cases} q_n \neq 0 \\ t_n^c \leq t_n^d + \Delta \\ \varphi_m(\pi) = 1, \forall p_m \in A_n \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

The overall benefit Ω of the streaming strategy $\pi \in \Pi$, that is equivalent to the quality perceived by the receiver, can now simply be expressed as the sum of the weights ω_n of all successfully decoded packets. We assume that packets whose $\varphi_n(\pi) \neq 1$ are simply discarded at the client, hence the overall benefit can be written as :

$$\Omega(\pi) = \sum_{\forall n:\varphi_n(\pi)=1} \omega_n.$$

C. Distortion Optimization Problem

Given the network assumptions and the abstraction of the encoded video bitstream, the distortion optimization problem consists in an efficient selection of the subset of video packets to be transmitted, jointly with their streaming policy. We assume a server-driven scenario in which the server is aware of, or can predict using a simple mechanism, the network conditions ($r_i(t)$ and $d_i(t)$), at each time instant. Given the deterministic packet transmission process over the network, the server will only schedule for transmission packets that can arrive at the client before their decoding deadline. Note that, in this scenario, the server needs at most one transmission attempt per packet.

The distortion optimization problem can be stated as follows: *Given* P, the packetized bitstream of an encoded video sequence, the maximum playback delay Δ imposed by the client, and the network state, ***find*** the optimal transmission policy $\pi^* \in \Pi$ that maximizes the overall quality measure Ω. The optimization problem translates into finding $\pi^* \in \Pi$ s.t.:

$$\Omega(\pi^*) = \max_{\pi \in \Pi} \sum_{\forall n:\varphi_n(\pi)=1} \omega_n.$$

The optimization problem can be easily reduced to the more general case of optimal scheduling problems. This family of problems proves to be NP-complete [2] and an optimal algorithm that solves them in polynomial time does not exist. In the remainder of this paper, we present an optimal algorithm that efficiently finds the distortion minimal streaming strategy for long video sequences and we propose a new heuristic algorithm that provides a close to optimal solution in polynomial time. Later we adapt our solution to support real-time streaming and we implement our solutions along with prefetch window mechanisms.

III. Packet Scheduling Analysis

A. Unlimited Buffer Nodes

This section proposes an analysis of the scheduling of packets in the streaming model described above, and computes the parameters necessary to solve the distortion optimization problem. We consider first the case where buffering space in the network nodes and the client is not constrained, i.e., $B_a = B_b = B_c = \infty$. The server has the knowledge of N video packets, where N can be the total number of network packets of the video stream (in the case of stored video), or simply the number of packets contained in the prefetch window in real-time streaming. The server is able to transmit network packets simultaneously on the two network paths. Under the assumption of unlimited buffer space, the server can send packets on each of the paths at the maximum rates of the first segments ($r_1(t)$ for path *a* or $r_3(t)$ for path *b*, see Figure 1).

Under a given policy π, the sending time t_n^s of each packet p_n can thus be easily computed. Suppose that p_n is sent on path *a* (i.e., $q_n = 1$). Let :

$$S_n^a(\pi) = \sum_{q_m=1; m<n} s_m$$

where $S_n^a(\pi)$ represents the cumulative size of all the packets that need to be sent on path *a* before p_n, under the policy π.

Under the assumption that the available bandwidth is fully utilized by the streaming application, t_n^s is the shortest time t at which $R_1(t)$ is larger than S_n^a :

$$t_n^s(\pi) = \arg \min_t |R_1(t) - S_n^a(\pi)|. \quad (1)$$

In other words, packet p_n can only be sent when all the previous packets scheduled on the same path have been transmitted.

Packet p_n will then arrive at the client after a certain delay, caused by the transmission delays (t_n^1 and t_n^2) on the 2 segments that compose path *a*, the latencies introduced by the

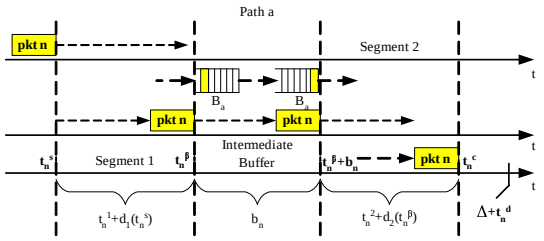


Fig. 3. Time diagram for packet p_n sent on path a.

two links ($d_1(t)$ and $d_2(t)$) and the queuing time at the node b_n. Therefore, the time instant at which packet p_n enters the node buffer can be expressed as :

$$t_n^\beta = t_n^s + t_n^1 + d_1(t_n^s).$$

Then, the arrival time of packet p_n at the client, can be written as :

$$t_n^c = t_n^\beta + b_n + t_n^2 + d_2(t_n^\beta).$$

The timing representation of the transmission of packet p_n is provided in Figure 3.

The transmission delays t_n^1 and t_n^2 represent the time needed to send packet p_n, at the available bandwidth. They have to verify the following relation :

$$R_1(t_n^s + t_n^1) - R_1(t_n^s) = R_2(t_n^\beta + t_n^2 + b_n) - R_2(t_n^\beta + b_n) = s_n,$$

and can be computed similarly to Eq. (1). The queuing time b_n corresponds to the time needed to transmit the $B(t_n^\beta)$ bits present in the buffer, at time t_n^β when packet p_n enters the buffer. The buffer fullness can be computed recursively as :

$$B(t_n^\beta) = \max[B(t_{n-1}^\beta) + s_{n-1} - R_2(t_n^\beta) + R_2(t_{n-1}^\beta),\ 0]$$

Therefore, the queuing time can be computed such that it satisfies :

$$R_2(t_n^\beta + b_n) - R_2(t_n^\beta) = B(t_n^\beta),$$

Note that, even if the previous development only consider the path *a*, the extension of the analysis to the packets transmitted over path *b* is straightforward.

The arrival time of packet p_n, t_n^c is thus fully determined. The playback delay $D(\pi)$ induced by the transmission policy π can finally be expressed as:

$$D(\pi) = \max_{1 \le n \le N}(D_n(\pi)) = \max_{1 \le n \le N}(t_n^c - t_n^d),$$

where $D_n(\pi)$ is the playback delay imposed by the streaming process up to packet p_n by the transmission policy π. An interesting property can be observed in the behavior of $D_n(\pi)$, that will be advantageously used in the scheduling optimization problem.

Lemma 1: Given that the streaming server sends the N network packets in parallel on two paths, and that on each path the packets are sent sequentially, the playback delay $D_n(\pi)$ under the given policy vector π is a non-decreasing function of n.

Proof: [Sketch] Observe that $D_n(\pi)$ can be expressed as a recursive function of n:

$$D_n(\pi) = \max(D_{n-1}(\pi), t_n^c - t_n^d) \tag{2}$$

Hence, $D_n(\pi) \ge D_i(\pi), \forall n \le N, \forall i \le n$, with: $D_0(\pi) = 0$ and $D(\pi) = D_N(\pi)$. ■

Let finally define the cumulative quality $\Omega_n(\pi)$, resulting from the streaming policy π. Starting from the quality of a perfect transmission where P is entirely transmitted, $\Omega_n(\pi)$ is decremented each time a packet is dropped. The cumulative quality, used later in the development of our optimal streaming algorithm, can be written as :

$$\Omega_n(\pi) = \begin{cases} \Omega_{n-1}(\pi) & \text{if } \varphi_n(\pi) = 1 \\ \Omega_{n-1}(\pi) - \omega_n & \text{otherwise} \end{cases} \tag{3}$$

with: $\Omega_0(\pi) = \sum_{n=1}^{N} \omega_n$ and $\Omega(\pi) = \Omega_N(\pi)$.

Lemma 2: Ω_n is a non-increasing function with packet number n.

Proof: [Sketch] Observe that ω_n is by definition a non negative value. Hence, $\Omega_n \le \Omega_i, \forall n \le N, \forall i \le n$. ■

While Eq. (3) does not explicit the influence of other packets that have packet p_n as their ancestor, the status $\varphi_n(\pi)$ of packet p_n, directly affects the status of all packets dependent on p_n.

B. Constrained Buffer Nodes

A similar timing analysis can be performed in the case where the buffering space in the intermediate nodes on each path is limited to B_a and B_b respectively. Without loss of generality, assume that the buffering space is larger than any video packet in P. B_a and B_b represent the buffer sizes allocated by the intermediate nodes to the streaming process and they are known by the server. There is no further feedback on buffer occupancy from the network nodes during the streaming process. In this case, the server tries to avoid buffer overflows, and needs to adapt the sending time of each packets, to the buffer fullness. It may no longer use the full available bandwidth, without risking to loose packets.

The streaming policy has to take into account these new constraints. In particular, if packet p_n has to be transmitted on path *a* under policy π, its sending time t_n^s is such that there is enough buffer space available when it reaches the intermediate node. Additionally, the packet p_n can only be sent when all the previous packets on the same path have been transmitted. Using the same notation as defined hereabove, t_n^s becomes the smallest value that simultaneously verifies the following conditions :

$$\begin{cases} R_1(t_n^s) \ge S_n^a(\pi) \\ t_n^s + t_n^1 + d_1(t_n^s) \ge \tau_n \end{cases} \tag{4}$$

where τ_n represents the earliest time at which there is enough space in the intermediate buffer to receive packet p_n. Equivalently, τ_n can be computed recursively since it verifies :

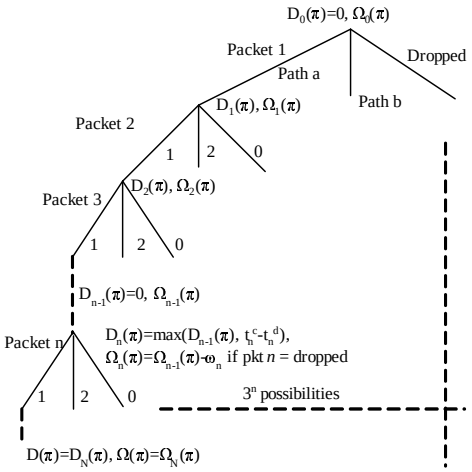


Fig. 4. Depth First Branch & Bound Algorithm

$$B_a - (B(t_{n-1}^{\beta}) + s_{n-1} - R_2(\tau_n) + R_2(t_{n-1}^{\beta})) \geq s_n.$$

We can also define the maximum buffer occupancy during the whole streaming process as:

$$B_a^{max}(\pi) = \max_{1 \leq i \leq N} (B(t_i^{\beta})) \leq B_a.$$

The timing analysis on path *b* follows immediately. The strategy π is thus completely defined, and we can compute $D(\pi)$ and $\Omega(\pi)$ similarly to the case of unlimited buffers.

In the multipath streaming scenario, the buffer capacities on the two disjoint paths may significantly influence the optimal packet scheduling, contrarily to the single path case. Note that a similar reasoning can be applied in order to prevent buffer overflow at the client, in case the client also has a limited storage space.

IV. Distortion Optimized Streaming

A. Optimal Solution: Depth-First Branch & Bound (B&B)

We first present an efficient algorithm that finds the optimal transmission policy vector π^* for a given encoded video sequence, network topology and playback delay. Since the sending and arrival times for each packet p_n can be computed for a given transmission policy π (see Section III), we can now search for the optimal packet scheduling π^* that maximizes the client video quality given an imposed playback delay. The optimization problem belongs to the larger set of scheduling problems that have a combinatorial complexity, hence, is NP-complete. An optimal polynomial time algorithm that can solve this problem is not known. However, efficient methods to solve our optimization problem exist. Our optimal solution is based on depth-first branch and bound (B&B) techniques [5].

The scheduling of N packets on two available paths can be organized as a decision tree of depth N (Figure 4). At each stage n in the tree, packet p_n can be sent on path a, on path b, or can be dropped. Hence, at depth N, the decision tree will contain 3^N leaves, according to the number of scheduling possibilities of the N packets on the 2 paths. At each stage n in the tree we can compute $D_n(\pi)$, the minimum playback delay and $\Omega_n(\pi)$, the cumulative video quality measure, for a partial scheduling up to packet p_n, according to the recursive Eq. (2) and Eq. (3) presented in Section III. This computation can be done for each one of the valid scheduling policies, for the first n packets. As mentioned in Section III-A, $D_n(\pi)$ and $\Omega_n(\pi)$ are non-decreasing, and respectively non-increasing functions in n. These two functions are used to establish a fast search on the decision tree for the optimal transmission policy vector π^*. A depth-first search is performed on the decision tree, starting with an initial policy vector π that satisfies the delay constraint $D(\pi) \leq \Delta$, where Δ is the playback delay imposed by the client. The policy π becomes our initial optimal policy π^* with $\Omega^* = \Omega(\pi^*)$. It is computed using a simple Earliest Deadline First algorithm with a complexity of $O(N)$, similar to [6], and adapted for media streaming applications (`Algorithm 1`). The EDF algorithm schedules frames in FIFO order. Packets belonging to a given frame are scheduled according to their importance ω_n, on the path that guarantees the earliest arrival time at the client. If a packet cannot be successfully scheduled, it is dropped without transmission, along with all his children packets, to avoid waste of network resources.

Algorithm 1 EDF Algorithm for computing the initial π^*

Input: N, $s_n, 1 \leq n \leq N$, $r_i(t), d_i(t), 1 \leq i \leq 4$,
Output: greedy scheduling policy π, $\Omega(\pi)$
Initialization: $t_1^s = 0$
for $n = 1$ to N **do**
  compute t_n^c(path a) and t_n^c(path b);
  if (t_n^c(path a) $\leq \Delta + t_n^d$ and t_n^c(path a) $\leq t_n^c$(path b)) **then**
    $q_n = 1$;
  else
    if t_n^c(path b) $\leq \Delta + t_n^c$ **then**
      $q_n = 2$;
    else
      $q_n = 0$;
    end if
  end if
end for
construct the scheduling policy vector $\pi = \{\pi_1, \pi_2, ...\pi_N\}$;
compute $\Omega(\pi)$.

Once we have the initial optimal policy and the initial benefit Ω^*, we start searching the decision tree for better transmission policies. We start with the leftmost transmission policy represented on the tree (equivalent to sending all packets on path a) and move through the decision tree towards right.

For each considered policy π', we compute $D_n(\pi')$ and $\Omega_n(\pi')$ successively for $n = 1...N$. At any packet p_n for which $D_n(\pi') > \Delta$ or $\Omega_n(\pi') \leq \Omega^*$, the computation of $D_n(\pi')$ is stopped, and the decision tree is pruned for all policies that have the same scheduling up to packet p_n (i.e., $\{\pi\}$ s.t. $\pi_i = \pi'_i$, $\forall i$, $1 \leq i \leq n$). If $D_N(\pi') \leq \Delta$ and $\Omega(\pi') \geq \Omega^*$, the policy π' becomes the new optimal policy π^* and $\Omega^* = \Omega(\pi')$. The operation is repeated until the set of all feasible policies Π represented on the decision tree has been covered. When the search is complete, the optimal policy π^* maximizes the video quality at the receiver and fulfills the playback delay constraints.

The B&B method provides an efficient way of computing

the optimal transmission policy vector π^*. The speed of the method depends on the pruning efficiency, which in turn, depends on the quality of the initial policy. However, the method is not scalable with N, since it cannot compute the optimal solution in polynomial time. The worst case complexity of the method remains $O(3^N)$. Also in the more general case of K independent network paths between the streaming server and the client, the complexity grows to $O((K+1)^N)$.

B. Heuristic Solution: Load Balancing Algorithm (LBA))

Since the B&B algorithm may be too complex in practice, this subsection presents a heuristic approach to find a close to optimal solution, in polynomial time. The algorithm is inspired from load balancing techniques, which offer close to optimal results in polynomial time for the problem of task scheduling in multiprocessor systems [7]. 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 value. Hence, we obtain a new representation of the encoded bitstream, $P' = \{p'_1, p'_2, ..., p'_N\}$, such that: $\omega_1(p'_1) \geq \omega_2(p'_2) \geq ... \geq \omega_N(p'_N)$. Then, a greedy algorithm (see `Algorithm 2`), similar to the EDF algorithm, schedules the N ordered packets on the two network paths, taking care of the packet interdependencies. On each of the two paths, a new packet is scheduled, and the packets are reordered according to their decoding deadlines, only if all other packets p' already scheduled on the paths can still meet their deadlines at the client. In the case where the value of each network packet is directly proportional to the size of the packet, the algorithm offers a real load balancing solution for the two network paths. `Algorithm 2` presents the sketch of the complete algorithm, where, for the sake of clarity, we redefine the action imposed on packet p'_n, q'_n as:

$$q'_n = \begin{cases} a & \text{if packet } p'_n \text{ is sent on } \textit{path a}; \\ b & \text{if packet } p'_n \text{ is sent on } \textit{path b}; \\ 0 & \text{if packet } p'_n \text{ is dropped without sending}; \\ \infty & \text{if packet } p'_n \text{ is not scheduled yet.} \end{cases}$$

To decide which action to take on each packet p'_n, 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 packet p'_n. This ensures that our algorithm does not waste network resources on transmitting network packets that cannot be correctly decoded at the receiver.

All packets marked to be scheduled on a given path, are reordered according to their decoding deadlines before transmission. When a new packet is inserted, it triggers a new packet ordering. If a packet p'_n can be scheduled on both network paths without interfering with the packets already scheduled, the algorithm will chose the path that offers the shortest arrival time for packet p'_n. If packet p'_n can only be scheduled on one path, the algorithm will insert the packet on that path. Otherwise packet p'_n cannot be scheduled on any of the two paths, without interfering with the already scheduled packets, and the algorithm will drop packet p'_n without transmitting it. Hence, the algorithm prevents that the transmission of one packet forces the loss of a more important packet previously scheduled because it arrives past its decoding time at the client.

Algorithm 2 Load Balancing Algorithm (LBA) for finding π

Input: P, ω_n, s_n, $1 \leq n \leq N$
Output: Suboptimal transmission policy vector π;
Initialization: Create P': arrange packets in order of importance ω_n; $n := 1$;
while $n \leq N$ **do**
  if Packet p'_n s.t. $q'_n = \infty$ **then**
    invoke Schedule_Packet(n);
  end if
  $n := n + 1$;
end while
Procedure: Schedule_Packet(n)
for all packets p'_k in A_n s.t. $q'_k = \infty$ **do**
  invoke Schedule_Packet(k);
end for
invoke do_Schedule(n);
Procedure: do_Schedule(n)
if $\exists$ packet $p'_k \in A_n$ s.t. $q'_k = 0$ **then**
  $q'_n = 0$;
  return;
else
  attempt the insertion of packet p'_n on path a and on path b, ordered according to the decoding deadlines, without compromising the decoding of any other scheduled packet;
  if t^c_n(path a), t^c_n(path b) $\leq t^d_n + \Delta$ **then**
    choose the path with shorter t^c_n;
    set q'_n accordingly;
  else
    if t^c_n(path a), t^c_n(path b) $> t^d_n + \Delta$ **then**
      $q'_n = 0$;
    else
      schedule packet p'_n on the path with $t^c_n \leq t^d_n + \Delta$;
      set q'_n accordingly;
    end if
  end if
end if

`Algorithm 2` performs an initial ordering of the N packets in the new set P'. Any common sorting algorithm that works with complexity $O(N \log N)$ can be employed. Afterwards, for each packet p'_n that must be scheduled, the algorithm requires a search among the packets already scheduled on each of the paths, in order to insert the new packet according to its decoding deadline. The operation requires $O(N)$ computations and is repeated N times, for each packet in P'. The complexity of the proposed algorithm is thus $O(N^2)$. For the more general case of K disjoint paths between the server and the client, the algorithm requires the computation of arrival times on all the paths, for all scheduled packets. The insertion of one packet therefore requires $O(KN)$ operations, and is performed for all N packets. The total complexity of `Algorithm 2` grows linearly with the number of network paths, being of $O(KN^2)$.

In conclusion, the proposed heuristic algorithm for finding a close to optimal transmission policy vector π has a complexity that grows linearly with the number of network paths K, and quadratic with the number of video packets N. Its low complexity makes `Algorithm 2` a suitable solution for fast

multipath packet scheduling, especially beneficial in real-time video streaming.

C. Real-time streaming: Sliding Window Approach

We now present the adaptation of the hereabove algorithms in the case of live streaming. In this case, the server does not anymore have the knowledge of the complete video sequence. Instead it receives the network packets directly from an encoder. The server may buffer live streams for δ seconds, in order to increase the scheduling efficiency. It has therefore a limited horizon, that we call the prefetch time δ. In other words, the prefetch time refers to the look-ahead window employed by the server. At any given time t, the server is therefore aware only of the network packets p_n whose decoding deadlines $t_n^d \leq t + \delta$.

We assume that $N(t)$ is the number of packets that are available at the server at time t, and that $P(t) = \{p_1, p_2, ..p_{N(t)}\}$ now represents the set of these packets ordered according to their decoding deadlines. $N(t)$ is equal to the number of packets containing data from the video sequence up to time $t + \delta$, minus the packets that were already transmitted to the client in the time interval $[0, t]$. $N(t)$ represents the size of the sliding window available at the server at time t.

The previously defined B&B and LBA methods are now applied on the set $P(t)$ in order to compute a transmission policy vector π for the $N(t)$ packets under consideration at time t. Neglecting the computation time, even for the B&B method, we can start transmitting the packets on the two paths according to the policy π, at time t. Let T be the time interval between two successive video frames, and without loss of generality, let t and δ be multiples of T. Hence, $t + \delta = kT$. At time t, the server can send packets that contain data from the encoded video sequence up to frame k. At time $t+T$, the packets containing data from frame $k + 1$ will be available at the server. At this time, the server will stop the transmission process of all packets from the previous sliding window that have not been sent yet, and add them to the new sliding window, along with the new packets from frame $k + 1$. B&B and the LBA methods are then applied on the new sliding window.

The implementation of our algorithms on top of a sliding window mechanism adapts the scheduling to new packets, as soon as they are available at the server. We compare the performance of this solution with the performance of the previous long horizon mechanisms for different video sequences in Section V.

It is worth mentioning, that in the case of real-time video streaming, `Algorithm 2` is equivalent to a sequential greedy packet scheduling algorithm that considers first the most important packets in the sliding window, while for a sliding window of just one frame, our LBA method reduces to the EDF algorithm. This observation emphasizes the low complexity of our proposal.

V. Simulation Results

A. Simulation Setup

This section now presents and discusses the performance of the proposed scheduling algorithms, in stored video scenarios, as well as in the case of realtime streaming. Video sequences are compressed with an MPEG4-FGS [8] encoder, at 30 fps. The chosen encoding format imposes I frames every 5 frames, and alternates P and B frames between successive I frames. The large frequency of I frames compared to usual encoding formats is mainly due to the complexity limitations of the B&B algorithm, that we use to find the optimal streaming solution. We use two different CIF video sequences, *foreman* and *news*, whose base layers are encoded respectively at $300kbps$ and $450kbps$, and enhancement FGS layers are encoded at $550kbps$. Each encoded frame is split into two network packets, one containing the data referring to the base layer, and one for the FGS layer information.

We set the weights ω_n of the video packets, as a function of their important relatively to the encoded bitstream. For example, the base layer packets generally represent the most important part of the information. In a first approximation, we choose the following packets weights: 5 for I frame base layer packet, 4 for the base layer of the first P frame, 3 for the base layer of the second P frame, 2 for the base layer of B frames, and 1 for enhancement layer packets. In general, the optimal transmission policy π^* is thus the strategy that successfully schedules the whole base layer of the video sequence, and the largest part of the enhancement layer.

Finally, the network latencies are neglected in the simulations (i.e., $d_i(t) = 0, \forall i, \forall t$), and the available bandwidth on the network segments are kept constant for the duration of the whole sequence (i.e., $r_i(t) = r_i$). Also we concentrate on the most likely scenarios where buffering is not constrained for the streaming applications.

B. Stored video scenarios

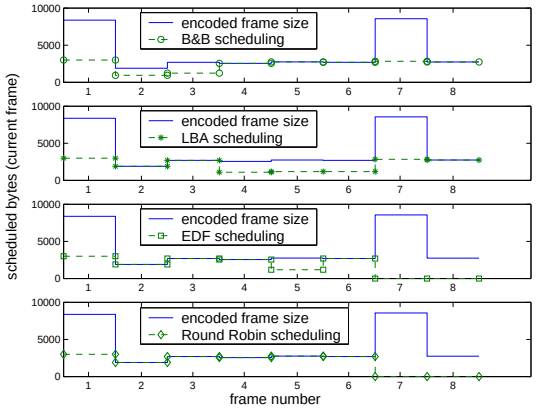


Fig. 5. Packet scheduling obtained by the B&B, LBA, EDF, and simple round robin algorithms for an IBPBPBIB frame sequence ($foreman\_cif$ sequence).

The proposed algorithms are first compared in the case of stored video scenarios, where the whole sequence is available at the streaming server before running the scheduling algorithms. Figure 5 presents the video rate trace at the decoder, when the server schedules the network packets according to the optimal B&B method, the LBA algorithm, the EDF algorithm [6], and a basic round robin strategy proportional to the

bandwidth on the two network paths. The segment bandwidth are set to $r_1 = 300kbps$, $r_2 = 500kbps$, $r_3 = 400kbps$ and $r_4 = 100kbps$ and the maximum playback delay imposed by the client is set to $\Delta = 150ms$.

It can be observed that, while the proposed LBA algorithm manages to successfully schedule almost the same number of packets as the optimal B&B solution, the simple EDF algorithm and the round robin method have clearly worse performance. This is due to the fact that the proposed LBA algorithm first schedules the most important packets (the packets from the base layer starting with the I frames, then P and B frames), and only later adds the enhancement layer packets. On the contrary, the EDF or round robin algorithms schedule as much as possible from any frame, without taking into account future frames. In this way, entire GOPs could be lost, because packets of the I frame cannot be successfully scheduled.

TABLE I

HEURISTIC ALGORITHMS PERFORMANCE COMPARISON

r_1	r_2	r_3	r_4	B&B	LBA	EDF
250	700	100	400	51.88%	47.03%	39.77%
300	700	100	400	58.90%	51.52%	43.44%
250	700	200	400	66.66%	60.65%	48.29%
250	700	250	400	68.26%	60.65%	48.29%
300	700	300	400	88.03%	82.24%	82.24%

A different representation is provided in Table I. It presents the performance of the LBA and EDF algorithms compared to the optimal solution for the $foreman\_cif$ sequence. The performance here is measured in terms of the percentage of successfully scheduled data bytes out of the total encoded stream. We observe that for a large variety of rates, the proposed LBA algorithm performs much closer to the optimal than the simple EDF approach. The largest loss in performance generally happen when the segments bandwidth are either similar, or very different.

C. Real-time video streaming

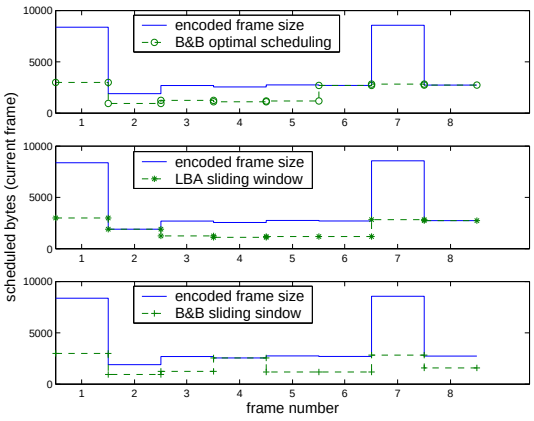


Fig. 6. Packet scheduling obtained by the B&B and LBA methods with sliding window, compared to the optimal scheduling for an IBPBPBIB frame sequence ($foreman\_cif$ sequence).

The proposed solution are now compared in the case of real-time video streaming, where the server knowledge is limited to the packets within the prefetch window. The prefetch window is set to 3 frames, and the maximal playback delay is $\Delta = 100ms$. Figure 6 compares the realtime adapted B&B and the LBA methods, where the original algorithms are applied on top of a sliding window mechanism. The performance of the optimal B&B method applied to the whole sequence is also provided for the sake of completeness. It can be seen that the B&B method is no longer optimal when combined with a sliding window, as expected. The proposed LBA algorithm can provide better performance.

The algorithms are also compared in terms of the proportion of transmitted information, for different network conditions, in Table II. Interestingly enough, the realtime LBA algorithm has a similar performance to the case of stored video scenario. The sliding window, even with low prefetch time, does not significantly influence the behavior of the scheduling algorithm. This property, along with the low complexity of the algorithm, proves that LBA represents a valid solution to multi-path packet scheduling in the case of real-time streaming.

TABLE II

ALGORITHM COMPARISON WITH SLIDING WINDOW

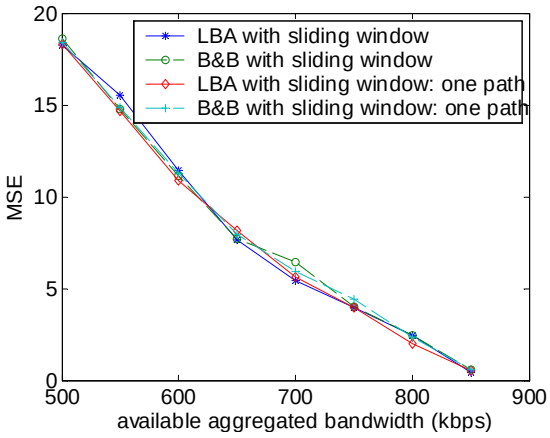
r_1	r_2	r_3	r_4	B&B	LBA	B&B SW	LBA SW
200	700	400	700	75.85%	65.55%	70.48%	65.55%
300	700	100	700	50.68%	47.03%	44.98%	47.03%
300	700	200	700	64.05%	60.87%	60.65%	60.87%
250	700	200	700	57.69%	51.46%	56.14%	51.52%
300	700	250	700	71.01%	60.87%	69.73%	60.87%

The algorithms are also compared in terms of the MSE perceived at the receiver. Figure 7 presents the distortion due to the network constraints, computed between the original encoded video sequence and the sequence available to the client. The MSE values obtained by the realtime B&B and LBA scheduling algorithms on two paths are compared to the ones obtained by using a single network path with equivalent aggregated bandwidth. Both schemes perform quite similarly when the aggregate bandwidth becomes large. We observe that there is virtually no loss in video quality when using two parallel network paths, instead of a single high bandwidth channel. This proves the efficiency of the proposed algorithms, relatively to the distortion lower-bound provided by the single channel scenario. Note that the EDF algorithm is voluntarily omitted here due to the high MSE values reached when it fails to schedule entire GOPs.

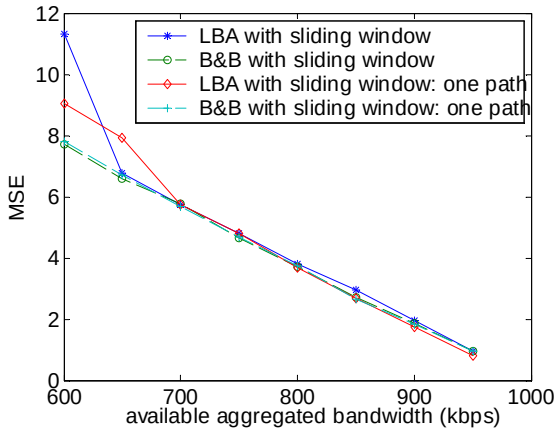
Finally, Figure 8 and Figure 9 illustrate the temporal behavior of the scheduling methods, when the minimal bandwidth on each path is set to $400kbps$ and $200kbps$ respectively. The instantaneous rate traces of the original encoded bitstream are presented, along with the traces of packets scheduled on both network paths, and rate of the reconstructed bitstream at the receiver. Both the B&B and LBA algorithms perform quite similarly in general, the rate variations on the paths are slightly smoother in the LBA method. Finally, it is not rare that frames are entirely sent on one path only.

D. Sliding Window Effect on Stream Scheduling

We now asses the influence of the size on the Sliding Window on the packet scheduling process. As seen before, in



(a) foreman_cif



(b) news_cif

Fig. 7. MSE values between the original encoded sequence and the scheduled one (100 frames)

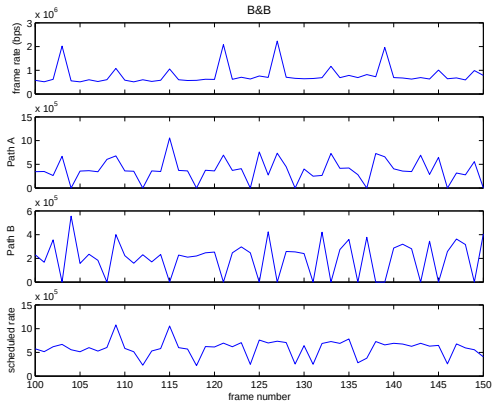


Fig. 8. Rate Diagram for B&B with sliding window for an aggregated bandwidth of $600kbps$, $foreman\_cif$ sequence

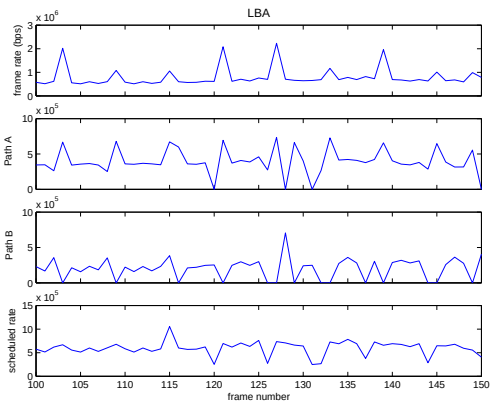


Fig. 9. Rate Diagram for LBA with sliding window for an aggregated bandwidth of $600kbps$, $foreman\_cif$ sequence

the case of constant link rates, the packet scheduling process is barely influenced by the size of the sliding window. However, it is not the case if we allow the link rates to vary in time.

We test the performance of the LBA algorithm given various sizes for the slinding window. We use the $foreman\_cif$ sequence (the first 100 frames) and the variable network rates as presented in Figure 10. We omit the results of the B&B algorithm due to the intractability of the computations for larger window sizes, and those of the EDF scheduling, since it is a simple FIFO scheduling that does not take into account the sliding window size.

The results of the LBA scheduling are presented in Figure 11. We can observe that for small sliding windows, the LBA algorithm performance is close the the one of the EDF algorithm, loosing entire GOPs. Results are improving once the sliding window increases, and the LBA algorithm has more flexibility in scheduling the video packets. Finally, given a reasonable sized window (half a second of prefetch), the results of the LBA are comparable to the performance in the case of whole sequence scheduling.

Table III presents the size of the scheduled bitstream (in percentage of the original bitstream) for various network rate sets and for various window sizes. While for the same rate set, the size of the scheduled bitstream does not vary with the window size, in terms of MSE, the differences are noticeable (Figure 12).

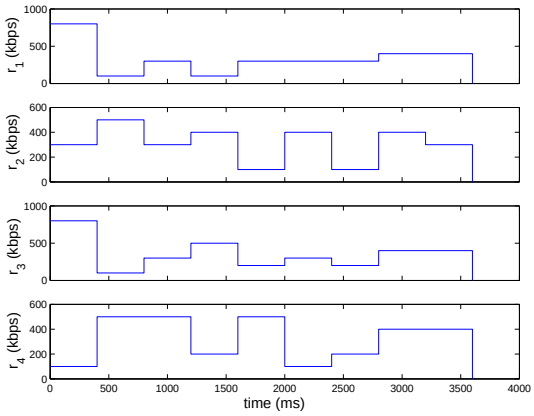


Fig. 10. Variation of network link rates in time

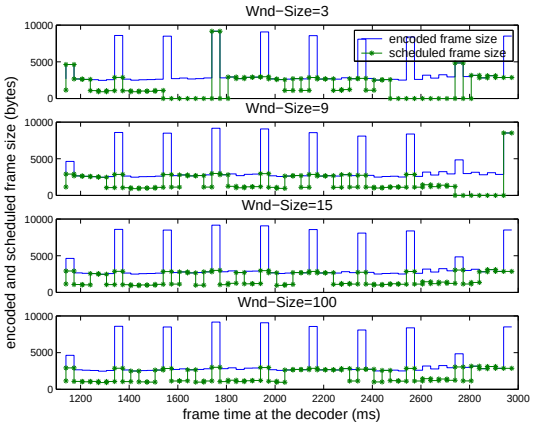


Fig. 11. LBA packet scheduling as a function of Sliding Window size

TABLE III

BITSTREAM SCHEDULED SIZE FOR DIFFERENT NETWORK RATE SETS, AS A FUNCTION OF SLIDING WINDOW SIZE

Window Size	SET 1	SET 2	SET 3	SET 4
3	57.33%	55.29%	64.54%	71.35%
5	57.14%	54.91%	64.63%	70.97%
7	57.03%	54.66%	64.53%	70.77%
9	56.21%	55.01%	64.53%	70.94%
11	57.62%	54.83%	64.48%	70.85%
13	57.38%	55.59%	64.61%	70.85%
15	57.41%	54.88%	64.26%	70.39%
20	57.80%	55.43%	64.46%	70.32%
100	57.29%	54.62%	63.97%	70.48%

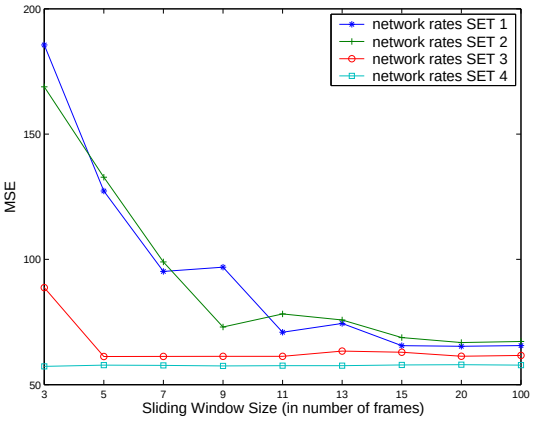


Fig. 12. MSE values for different network rate sets as a function of Sliding Window size

E. Constrained Buffer Nodes

We further investigate the effect on scheduling of the size of intermediate buffers.

For the same network rate set as in Figure 10, we vary the size of intermediate nodes buffers (B_a and B_b). Figure 13 presents the LBA scheduling for different buffer sizes. We observe that, for the same network rates, bigger intermediate buffers allow the scheduling of more video packets, smoothing better the rate variations.

Table IV presents similar results, in term of scheduled stream size, for multiple network rate sets. We observe differences of up to 10% in the scheduled streams sizes for the same rate set. The difference is also noticeable in terms of MSE (Figure 14).

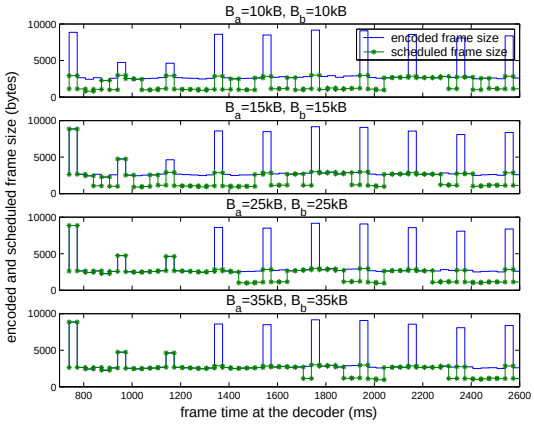


Fig. 13. LBA scheduling as a function of intermediate buffer size

TABLE IV

BITSTREAM SCHEDULED SIZE FOR DIFFERENT NETWORK RATE SETS, AS A FUNCTION OF INTERMEDIATE BUFFER SIZE (IN KB)

Buffer Size	SET 1	SET 2	SET 3	SET 4
10 kB	57.29%	55.10%	63.97%	73.26%
15 kB	60.17%	59.52%	67.55%	76.65%
20 kB	61.50%	60.00%	68.34%	78.78%
25 kB	63.13%	61.71%	69.61%	80.57%
30 kB	63.13%	62.71%	70.97%	81.74%
35 kB	63.13%	64.09%	70.98%	83.58%

Figure 15 shows the content of the two intermediate buffers during the streaming process, in case the two buffers are limited to $10kB$. We observe how the two buffers act as smoothers for the network rate variations on the two paths.

Finally, we study the effect of the intermediate buffer size on the packet load balancing on the two network paths. Figure 16 presents the load balancing of the video stream on the two network paths when there are no intermediate buffer constraints, while Figure 17 presents the same streaming scenario in case the intermediate buffer on the first network path is limited to $8kB$. We observe major differences in the packet scheduling on the two paths between the two scenarios.

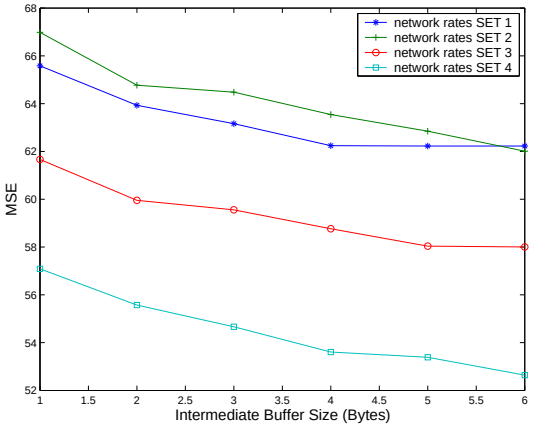


Fig. 14. MSE values for different network rate sets as a function of Intermediate Buffer size

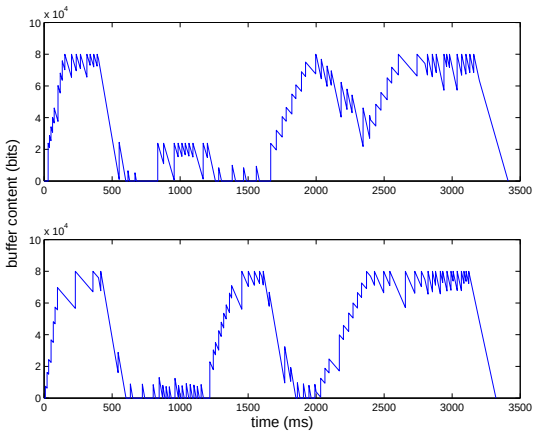


Fig. 15. Intermediate buffer content during the streaming process ($B_a = B_b = 10kB$)

A smaller buffer size on the first network path will render it unusable for a considerable period of time. While the shortage is partially compensated by sending more packets on the second link during the specific period, the effects on the received bitstream are noticeable. The scheduling of the bitstream in the case of unlimited intermediate buffers is smoother.

F. Rate Prediction

Next we test our proposed scheduling algorithm in the case of network rate prediction. The server is no longer aware of the available rates of the links composing the two path. IT knows only the expected average rate of all the links.

We model each of the intermediate network links as a discrete-time system, with a sampling interval of T_s seconds. Therefore, each link can communicate a maximum of $r_i T_s$ bits of data in the time interval $[iT_s, (i+1T_s)]$, where r_i is the available bandwidth of the given link in the i_{th} time step.

We model the process r_i as a Gaussian autoregressive process of the form: $r_i = \mu + (1-\alpha)\sum_{j=0}^{\infty} \alpha^j n_{i-j}$, $j \in Z$, $n_k = 0, \forall k < 0$, where each n_j is an independent zero mean Gaussian random variable with variance σ^2, μ denotes the average available bandwidth, and α is a modelling parameter [9].

For the rate prediction at the server, we use a simple auto-regressive prediction model, in which, the available bandwidth of a given link in the next time interval $i+1$ is given by: $r_{i+1} = a\frac{\sum_{j=1}^{i-1} r_j}{i-1} + br_i$, where a and b are the prediction coefficients satisfying $a + b = 1$.

In the following simulations we set all required parameters to the values presented in Table V.

These values insure a rate variation for al the links of up to 15% of the average value. We schedule the first 100 frames of $foreman_cif$ encoded into a base layer at $350kbps$ and an enhancement layer at $500kbps$. We compare the results obtained by our algorithm in the case the server knows the rates in advance with the case when it predicts the rates based on the average value and past instantaneous values. We set the maximum playback imposed by the client at $D = 200ms$.

TABLE V
PARAMETER VALUES FOR NETWORK PREDICTION

a	0.8
b	0.2
σ^2	50
α	0.8
T_s	$400ms$
μ_1, μ_2	$320kbps$
μ_3, μ_4	$250kbps$

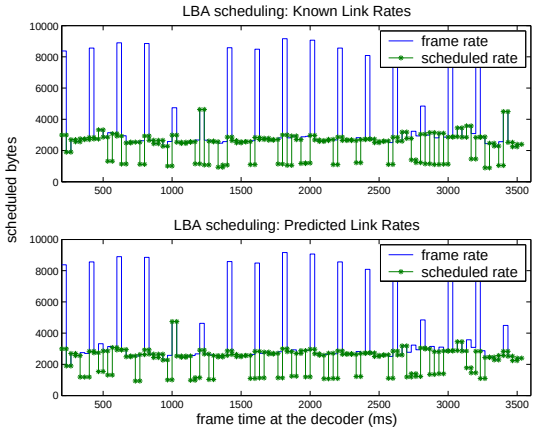


Fig. 18. LBA scheduling: real rates vs. predicted rates

For the scheduling based on predicted rates we use a delay of $D_1 = 150ms$, in order to cope with big shifts in link rates. The results are presented in Figure 18.

We observe that the performance degradation compared with the optimal case, when all rates are known, is negligible. While, in the optimal case, the algorithm correctly schedules 161 packets, out of 200, representing 66.96% of the total stream, in the case of prediction, it manages the correct schedule of 158 packets, representing 65.57%. We observe that no frame is lost due to frame dropping or late packet arrivals at the client. This gives a similar performance in terms of client video quality.

G. Complexity Issues

We investigate the complexity of the proposed algorithms and we try to derive an optimal functioning point for our LBA method in terms of complexity vs. performance.

Figure 19 presents the complexity of the proposed optimal B&B algorithm, our LBA algorithm, and the simple EDF algorithm as a function of the size of the sliding window used.

We observe that, as expected, the EDF algorithm has the lowest complexity, which does not depend of the size of the sliding window. Its complexity is linear with the number of frames scheduled, but in the same time, its performance is very low compared with the other algorithms. The B&B algorithm proves to have a prohibitive complexity. I is exponential in the number of frames schedules. For comparison reasons, we only present its complexity for a few scheduled frames.

Finally, the LBA algorithm presents a complexity that depends on the number of frames scheduled and the size of the sliding window used. Its complexity (C) varies according to the formula: $C = 2(Wnd\_Size)^2(N - Wnd\_Size)$.

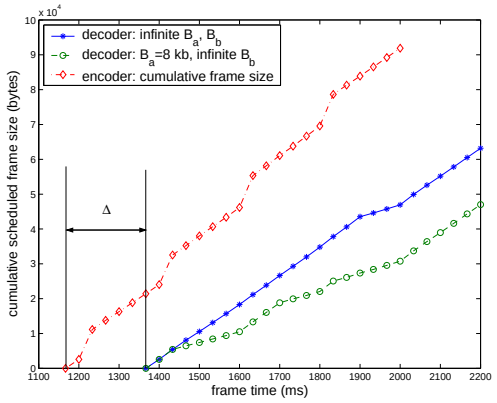


Fig. 16. LBA scheduling on the two paths without intermediate buffer constraints

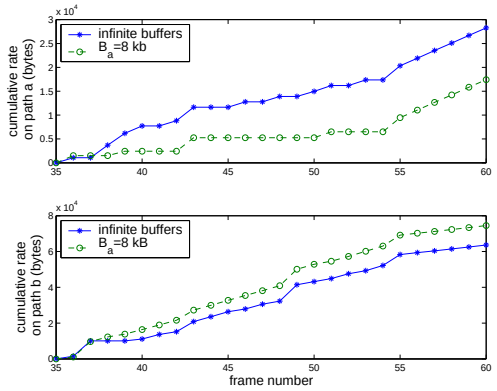


Fig. 17. LBA scheduling on the two paths with buffer constraints on path a ($B_a = 8kB$)

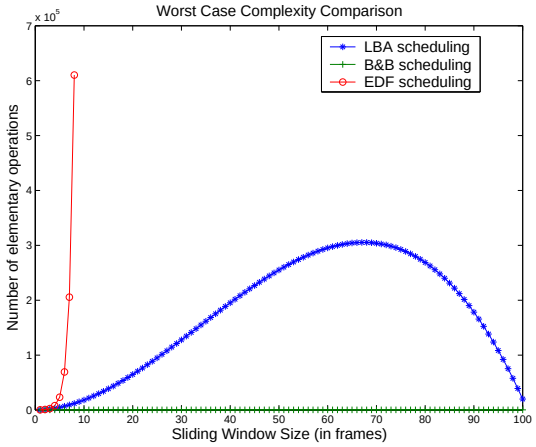


Fig. 19. B&B, LBA and EDF complexity as a function of the size of the Sliding Window

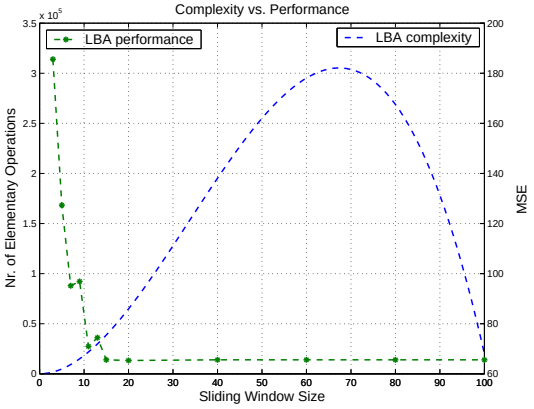


Fig. 20. LBA performance vs. complexity

Figure 20 presents the performance of the LBA algorithm for different sizes of the sliding window. We superimpose the complexity curve in order to find the optimum operation point of the algorithm as a function of performance and complexity. We observe that for quite low values for the sliding window, the performance of the LBA algorithm matches the one of on-demand scheduling when all frames are known in advance. For low values of the sliding window, the complexity of the LBA algorithm is also low.

Its low complexity and good performance, even for small sliding window sizes, make the LBA a suitable candidate for real time packet scheduling in multimedia streaming.

VI. Related Work

Multi-path video streaming has recently drawn the attention of the scientific community. The benefits of multi-path routing in multi-path media streaming are presented in [10] and [11]. Among the main benefits of using multiple paths between a media server and a client we enumerate: (i) the reduction in correlation between packet losses, (ii) increased throughput, and (iii) ability to adjust to variations of congestion patterns on different parts of the network. Ongoing research is directed towards solving problems associated with multipath streaming scenarios, as presented in [12].

An experimental approach on path-diversity is provided in [13]. The authors select the optimal pair of servers containing complementary video descriptions for each client while accounting for path lengths, jointness and disjointness. A receiver driven rate-distortion optimization framework with server diversity is presented in [14] and [15]. The authors solve a R-D problem in a Markov Decision Process (MDP) framework, for the case of multiple servers containing data from the same requested video stream. The problem of finding the optimal set of network paths between the server and the client, that ensures a minimum startup delay is solved in [16]. The authors of [17] present a path-diversity system with FEC for packet switched networks, while the authors of [18] compare multi-path streaming solutions implemented at the transport and application layer. Multi-path streaming solutions for wireless networks are proposed in [19], [20].

Our approach to multipath streaming is different than the previous work. We search for optimal transmission policies for sets of sequential video packets given the network scenario and the encoded bitstreams. By extensive simulations in which we consider variable network parameters, we emphasize the conceptual issues concerning the optimization problem, and we highlight the major differences between the proposed

solutions. The new framework proposed in this paper, permits the analysis of scheduling stored sequences as well as live streams.

Perhaps the closest existing work to our approach is the one presented in [6]. The authors present a heuristic algorithm for packet scheduling on multiple heterogenous networks. The algorithm performance is similar to the one of scheduling packets on a single network path with the same aggregated bandwidth and outperforms other algorithms that derive from round robin schemes. The presented Earliest Deadline First algorithm is adapted for media streaming scenarios in our work, and is used as basis of comparison along Round Robin schemes in our simulations. However, unlike the heuristic approach in [6], we derive an optimal approach to packet scheduling, and our heuristic approach outperforms the EDF algorithm in the case of stored video and live stream scheduling.

A single path optimal packet scheduling mechanism for multiple description coded video sequences is presented in [21]. In [22], the authors solve an optimization scheduling problem specific to wireless networks, using a partially observable MDP, while the authors of [23] propose an opportunistic traffic scheduling algorithm for multiple network paths. An analysis of optimal layered video streaming is also provided in [24].

Finally, unlike the work done in network striping [25] and traffic dispersion [26], our work implements packet scheduling algorithms on multiple network paths, towards the final goal of achieving an optimal video quality at the receiver. We do not only take advantage of the increased aggregated bandwidth of multiple network paths, but in the same time, we also use the different paths to reduce the client playback delay.

VII. Discussion and Conclusions

This work addresses the problem of the joint selection and scheduling of video packets on a network topology that offers multiple paths between the streaming server and the media client. We use an encoded video abstraction model that factors in the variable importance of video packets, as well as their interdependencies. An optimization problem is then formulated, that aims at maximizing the video quality at the client under a given playback delay. A formal analysis of packet transmission timing leads to the derivation of efficient algorithms to find the transmission policy that maximizes the video quality at the client. Because of the complexity of the optimal method, we propose fast, polynomial time algorithms that still offer close to optimal solutions. Both methods have been implemented in the case of stored videos, and real-time streaming with the help of a sliding window mechanism. Simulation results in both scenarios prove that our proposed heuristic-based solution performs well in terms of final video quality, and is moreover suitable for the case of real-time streaming under strict delay constraints. They also show that our methods outperform other common scheduling algorithms from the literature.

Our method can be easily adapted to network scenarios characterized by weaker assumptions in terms of server knowledge about link rates and loss processes. We show how the algorithms perform in the case of predicted network rates, when the server uses a simple auto-regressive prediction mechanism. By using more conservative scheduling parameters, our scheduling methods cope with large variations in instantaneous network rates, with a negligible increase in the client perceived distortion. Furthermore, link loss can be effectively addressed by implementing FEC schemes on top of our scheduling mechanisms. In our future work we will extend our analysis and experiments for the case of more complex network scenarios, and release the assumption of lossless packet transmission.

References

[1] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Proceedings of the Packet Video Workshop, Pittsburg, PA*, 2002.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*, 23rd ed., V. Klee, Ed. W. H. Freeman and Company, 2002.

[3] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, 2nd ed. Springer-Verlag, 2000.

[4] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *submitted to IEEE Transactions on Multimedia*, 2001.

[5] J. Jonsson and K. G. Shin, "A parametrized branch and bound strategy for scheduling precedence-constrained tasks on a multiprocessor system," in *Proceedings of IEEE ICPP*, 11-25 August 1997, pp. 158–165.

[6] K. Chebrolu and R. Rao, "Bandwidth aggregation in heterogenous networks," 2003, http://adaptive.ucsd.edu/downloads.htm.

[7] F. A. Samadzadeh and G. E. Hedrick, "Near-optimal multiprocessor scheduling," in *Proceedings of ACM Computer Science Conference*, April 1992, pp. 477 – 484.

[8] Radha H.M., van der Schaar M. and Chen Y., "The mpeg-4 fine-grained scalable video coding method for multimedia streaming over ip," *Transactions on Multimedia*, vol. 3, no. 1, pp. 53–68, March 2001.

[9] A. Sehgal, O. Verscheure, and P. Frossard, "Distortion-buffer optimized tcp video streaming," in *Proceedings of IEEE ICIP*, 2004.

[10] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-path vs. multi-path overlay routing," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurements*, October 2003, pp. 91–100.

[11] L. Golubchik, J. Lui, T. Tung, A. Chow, and W. Lee, "Multi-path continuous media streaming: What are the benefits?" *ACM Journal of Performance Evaluation*, vol. 49, no. 1-4, pp. 429–449, Sept 2002.

[12] J. G. Apostolopoulos and M. D. Trott, "Path diversity for enhanced media streaming," *IEEE Communications Magazine*, vol. 42, no. 8, pp. 80–87, August 2004.

[13] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proceedings of IEEE INFOCOM*, vol. 3, 23-27 June 2002, pp. 1736–1745.

[14] J. Chakareski and B. Girod, "Rate-distortion optimized packet scheduling and routing for media streaming with path diversity," in *Proceedings of the IEEE Data Compression Conference (DCC)*, 25-27 March 2003, pp. 203–212.

[15] ——, "Server diversity in rate-distortion optimized media streaming," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 3, 14-17 September 2003, pp. III – 645–8 vol.2.

[16] J. Chen and S.-H. Chan, "Multipath routing for video unicast over bandwidth-limited networks," in *Proceedings of IEEE Globecom*, vol. 3, 25-29 November 2001, pp. 1963–1967.

[17] T. Nguyen, P. Mehra, and A. Zakhor, "Path diversity and bandwidth allocation for multimedia streaming," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, vol. 1, 6-9 July 2003, pp. 1–4.

[18] R. Karrer and T. Gross, "Multipath streaming in best-effort netoworks," in *Proceedings of IEEE ICC*, vol. 2, 11-15 May 2003, pp. 901–907.

[19] H. Man and Y. Li, "A multipath video delivery scheme over diffserv wireless lans," in *Proceedings of SPIE-IS&T Electronic Imaging*, 2004.

[20] Y. Wang, S. Panwar, S. Lin, and S. Mao, "Wireless video transport using path diversity: Multiple description vs. layered coding," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, vol. 1, 22-25 September 2002, pp. 21–24.

[21] Y. C. Lee, Y. Altunbasak, and R. M. Mesereau, "Optimal packet scheduling for multiple description coded video transmissions over lossy networks," in *Proceedings of IEEE Globecom*, vol. 6, 1-5 December 2003, pp. 3569–3573.
[22] D. Tian, X. Li, G. Al-Regib, Y. Altunbasak, and J. Jackson, "Optimal packet scheduling for wireless streaming with error-prone feedback," in *Proceedings of IEEE WCNC*, vol. 2, 21-25 March 2004, pp. 1287–1292.
[23] C. Cetinkaya and E. W. Knightly, "Opportunistic traffic scheduling over multiple network paths," in *Proceedings of IEEE INFOCOM*, 2004.
[24] D. Saparilla and W. W. Ross, "Optimal streaming of layered video," in *Proceedings of IEEE INFOCOM*, vol. 2, 26-31 March 2000, pp. 737–746.
[25] C. Brendan, S. Traw, and J. M. Smith, "Striping within the network subsystem," *IEEE Network*, pp. 22–32, July/August 1995.
[26] E. Gustafsson and G. Karlsson, "A literature survey on traffic dispersion," *IEEE Network*, vol. 11, no. 2, pp. 28–36, March/April 1997.