Optimal video stream multiplexing through linear programming

Helman I. Stern\textsuperscript{a}, Ofer Hadar\textsuperscript{b,*}, Nir Friedman\textsuperscript{a}

\textsuperscript{a}Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, P.O. Box 653, Be'er-Sheva 84105, Israel
\textsuperscript{b}Department of Communication Systems Engineering, Ben-Gurion University of the Negev, P.O. Box 653, Be'er-Sheva 84105, Israel

Received 11 January 2007; received in revised form 15 November 2007; accepted 1 February 2008

Abstract

This paper presents a new optimal multiplexing scheme for compressed video streams based on their individual e-PCRTT transmission schedules. A linear programming algorithm is proffered, which takes into account the different constraints of each client. The algorithm simultaneously finds the optimum total multiplexed and individual stream schedules that minimize the peak transmission rate. Since the problem is formulated as a linear program it is bounded in polynomial time. It is shown that the algorithm succeeds in obtaining maximum bandwidth utilization with Quality of Service (QoS) guarantees. Simulation results using 10 real MPEG-1 video sequences are presented. The optimal multiplexing linear programming results are compared to the e-PCRTT and Join-the-Shortest-Queue (JSQ) procedures in terms of peak transmission bandwidth, P-loss performance and standard deviation. For several client buffer sizes, the rate obtained by our LP solution when compared to a previous e-PCRTT and JSQ methods resulted in reductions of 47% and 56%, respectively. This implies for a fixed rate problem that the proposed scheme can allow an increase in the number of simultaneously served video streams.

Keywords: Video rate smoothing; Optimal multiplexing; Statistical multiplexing gain; Network utilization; Linear programming; P-loss

1. Introduction

The main problem of transmitting compressed video over communication networks is its burstiness. Compressed movies exhibit peak rates that are often significantly larger than their long-term average rate. One of the suggested solutions for reducing burstiness is to use a playback buffer at the client side (PC, workstation, or set-top box). In such a system, the video streams are stored at a multimedia server and transmitted through the network to the client site. The client’s video card then decodes the stream and forwards it to the video display. The server can significantly reduce the bandwidth requirements for transmitting stored video streams by prefetching frames into the client playback buffer. In order to compute an efficient server transmission plan, bandwidth-smoothing algorithms usually require a priori knowledge of the client buffer size, and the length of the transmitted video frames. Many kinds of video smoothing techniques [2,21,3,5,1,19] have been developed for reducing the bandwidth variability of video streams in order to improve network utilization. These techniques exploit client buffering and determine a smooth rate transmission schedule.

\*Corresponding author.
E-mail addresses: helman@bgumail.bgu.ac.il (H.I. Stern), hadar@bgumail.bgu.ac.il (O. Hadar).
The traffic management schemes for variable bite rate (VBR) videos fall into four main categories [17]: (1) deterministic without smoothing, (2) deterministic with smoothing, (3) probabilistic, and (4) probabilistic with collaborative prefetching schemes. In the first scheme [13,14], the original VBR traffic is sent into the network. Hence, for highly variable VBR traffic this scheme requires large initial delays to achieve moderate link utilization. The second scheme [4,6], sends a smoothed version of the original VBR traffic into the network. The video traffic is transmitted at different constant rates over different intervals according to a specific transmission schedule. The piece-wise constant-rate transmission schedule takes advantage of the fact that some of the prerecorded video can be prefetched into the client buffer. The challenge lies in computing those transmission schedules that have the smallest peak rate and rate variability while ensuring that the client buffer neither overflows nor underflows. The probabilistic schemes send the original VBR encoded video into a non-buffered multiplexer [16,18]. This scheme introduces packets loss whenever the aggregate transmission rate exceeds the data link rate. The original traffic can be first smoothed before it is statistically multiplexed at a non-buffered link. The statistical multiplexing of the smoothed traffic increases link utilization at the expense of small packet loss probabilities. Reisslein and Ross [17] present a probabilistic transmission scheme with collaborative prefetching that determines the transmission schedule of an on-line connection as a function of the buffer contents at all the clients. It is shown that prefetching protocols with collaborative prefetching have less packet loss than the optimal smoothing algorithm [19] for the same link utilization.

Using smoothing, one can represent a VBR stream as a series of fixed constant bit rates (CBRs), thus simplifying the allocation of network resources and increasing their utilization. This is the approach used by the piecewise constant rate transmission and transport (PCRTT) algorithm found in [17]. The difficulty of the PCRTT approach is that it treats each stream independently thus providing a sub optimal solution. An enhancement to PCRTT, called e-PCRTT, will be used in this paper. In this paper, we retain the CBR assumption, and propose a linear programming method, which treats all video streams collectively to obtain an optimal solution. In [22] the optimal smoothing algorithm is considered together with statistical multiplexing. Their investigation considers performance when small packet-losses are allowed. This scheme will provide not only statistical Quality of Service (QoS) guarantees, but provides increased system utilization.

In parallel to smoothing algorithms there has been some work on developing schemes based on prefetching [16,18]. These protocols use VBR transmission, and can only provide statistical QoS guarantees. In [16], a protocol called Join-the-Shortest-Queue (JSQ) prefetching is presented which has many advantages compared to the other transmission schemes in the literature. JSQ achieves very high network utilization, facilitates user interactions, and has minimal start-up latency. Their experimental results showed that JSQ prefetching had a loss probability several orders of magnitude smaller than optimal smoothing [22] for the same buffer size and network utilization. Therefore, in this paper we compared our multiplexing results to the JSQ, which is considered to be the most advanced algorithm for video transmission multiplexing. Our purpose is to solve simultaneously the problem of individual smoothing and multiplexing of several video streams. Our method derives the transmission schedule for each individual stream by taking into account the interaction between streams, and optimizes the bandwidth allocation of the aggregated channel bandwidth. The main idea of the JSQ is to put a buffer in the client side, allowing prefetching of future frames when the transmission link is not fully utilized. The idea of video prefetching in JSQ is to keep the client’s buffers as full as possible at all times, so that in intervals where the aggregate bit-rate exceeds the link capacity the user will not experience playback starvation. In our protocol the buffer is used to reduce the peak rate and the rate variability of the video sequence, and in some time intervals it will be almost empty. Another interesting work [18] also utilizes smoothing along with prefetching, but for the case where all video servers are distributed in a network environment. That work is actually an enhancement of JSQ for distributed environment, and not a centralized scenario where there is only one server as in our case.

Our algorithm is planned to be implemented in a system where the program schedule is known in advance (prerecorded VBR sources), and not for real time situations such as Video on Demand (VoD). In our analysis we assume a network topology were all clients are physically separated but have the same buffer size. In our case, the
optimization is done for a static situation, where it is known in advance all video programs that need to be transmitted in a common channel with maximum efficiency. So the complexity of the algorithm is not a critical issue here, because we can apply our algorithm off line. However, the system can be adapted to operate in a dynamic mode. One needs only to revise the transmission plan whenever a new demand is received. In addition, all our performance comparisons are done against “off-line” systems.

The rest of the paper is organized as follows. In Section 2, the optimal multiplexing problem is defined. Section 3 presents the notation and the formal description of the problem. The conversion of the Multiplexed Problem to an upper bounded linear program (LP) is presented in Section 4. Section 5 deals with the influence of the total buffer size of all clients on the optimal solution, as well as a new LP formulation of the multiplexing optimal buffer allocation problem. Section 6 presents several small numerical examples using the proposed scheme. Section 7 provides LP solutions using ten real video streams. The LP, ePCRTT and JSQ solutions are compared using $P_{\text{loss}}$ in Sections 8, respectively. The effect of the number of movies, and the linearizing approximation in the LP scheme is discussed in Section 9. Section 10 concludes the paper.

2. The optimal multiplexing problem

Many researchers have tried to find the optimal solution for transmitting stored video streams using smoothing techniques that are applied to each individual stream [2,3,7–11,17]. However, most of smoothing schemes do not exploit the statistical multiplexing gain that can be obtained before smoothing. Kang and Yeom [12] proposed a scheme that deals not only with an individual stream, but also with the set of streams being serviced in order to achieve the optimal network bandwidth utilization and maximize the number of streams that can be serviced simultaneously...

2.1. The PCRTT and e-PCRTT smoothing algorithms

In this section a smoothing algorithm, which employs CBRs over a series of equal size intervals is described. This smoothing technique is called a piecewise constant rate transmission and transport (PCRTT) smoothing technique in the literature [17]. PCRTT determines individually for each video stream a transmission rate for each time interval. The rate at each interval is determined according to the intersection points of the vertical borderlines of the time intervals with the accumulative data curve, $L$ (see Section 3). These curves determine a piecewise linear function. The slope of the $j$th linear interval, over interval $j$, corresponds to a rate $r_j$ in a bandwidth transmission plan. To avoid buffer underflow, PCRTT offsets this plan vertically when needed. This guarantees that the whole run lies above the $L$ curve [17]. Raising the plan corresponds to introducing an initial playback delay at the client site. Since a PCRTT rate plan consists of fixed-size intervals, the bandwidth changes occur at constant times. In the PCRTT scheme any increase of the buffer size above a threshold value will not influence the results of the transmission schedule.

An enhancement to PCRTT, called e-PCRTT, has been proposed in [8]. The idea behind e-PCRTT is to impose local constraints on the trajectory bandwidth rate plan instead of the global constraints imposed by the original algorithm. e-PCRTT aims at reducing the required buffer space for the same fixed-size intervals. This algorithm can increase the length of the smoothing interval while using the same buffer size. The practical implication of increasing the interval size is that the number of bandwidth changes can be reduced. This is of crucial importance for ATM networks or reserved IP networks that allow users to renegotiate their traffic parameters.

2.2. Collective stream smoothing

The PCRTT and e-PCRTT smoothing schemes treat multiple video sources by smoothing the individual streams independently and not collectively. Thus, they only achieve the optimal solution for an individual stream but not an optimum solution for the multiplexed stream. In this paper, we propose a new smoothing scheme that deals with not only the individual streams, but also the entire complex of streams serviced simultaneously. We call this the Optimal Multiplexing Problem, and solve it using a linear programming formulation.

The advantage of our suggested methodology is not only that it attains the optimum solution, but also that it allows the possibility of imposing different constraints on each of the client’s buffer sizes. Using our linear programming scheme it is possible to simultaneously minimize the aggregated
peak-rate and variability of the multiplexed stream. Throughout the paper, we consider only the end nodes behavior, while assuming the network provides the QoS required by the algorithm, namely guaranteed bandwidth and no jitter. Hence, data loss is only attributed to decoder buffer overflow and not to network congestion.

There are many ways to determine the starting time of each multiplexed stream. Assuming the stream arrangement is achieved by enforcing the starting time of the multiplexed video streams, we describe a possible arrangement for multiplexing smoothed video streams. We constrain all the streams to be synchronized in terms of the start time of the first smoothing interval. By synchronizing the streams to be synchronized in terms of the start time of each multiplexed stream, we describe a possible arrangement for multiplexing smoothed video streams. We constrain all the streams to be synchronized in terms of the start time of the first smoothing interval. By synchronizing the streams, in a previous paper [9], it was shown that it is possible to bound the number of bandwidth changes in the multiplexed stream; and also to reduce the variance of the multiplexed stream.

In what follows, we present the globally optimized smoothing scheme, which also consists of individual smoothing using constant intervals, which maximizes the number of active streams while providing deterministic service to each client. The proposed scheme is based on a piecewise linear approximation of the accumulative data curve of each of the compressed video streams. This approximation allows the optimal multiplexing problem to be formulated as a LP. The suggested scheme minimizes the aggregated constraint of overall client buffer requirements.

2.3. JSQ, prefetching algorithm

The objective of video prefetching, is to keep the client buffers as full as possible at all time intervals. While doing so, the scheme strives to have all client’s buffers non-empty, which gives a scenario where there is no starvation. There is also, no overflow on the server bandwidth throughout the algorithm process. In testing our LP schema against the JSQ we have used an extended JSQ implementation which does not cease transmitting frames when it has no more bandwidth for the emptiest buffer; but it tries to deliver a frame to the second emptiest buffer, and so on, until its remaining bandwidth is so low it cannot deliver frames to any other client’s buffer. Using real video streams (Section 7) in the comparison test, the extended version of JSQ has been used because it has better smoothing performances, and because this version was simulated in the original paper [16].

3. Notation and problem description

Assume the frame rate is fixed, so that characterizing the variability of the bit rate amounts to modeling the frame size sequence. Consider \( F \) video streams, with a fixed frame-per-second (FPS) rate. Let \( F(t) \) be the \( r \)th frame in the sequence of the \( F \) video stream \( (v = 1, 2, \ldots, K) \). Frame \( F(v) \) is of size \( b(v) \) bits. Assume equal time units between successive frames (1/FPS). The cumulative data at time \( t \) is \( L(t) = \sum_{i=1}^t b(v) \). To prevent overflow in a buffer of size \( b \) bits, at time \( t \), the amount transmitted should not exceed \( U(t) = \sum_{i=1}^t b(v) + b = L(t) + b \).

Assume \( m \) video streams/clients, represented by the index \( j = 1, 2, \ldots, m \). Partition each stream into equal intervals of length \( \Delta t \). \( L_j(t) \) and \( U_j(t) \) represent the lower and upper banks of an imagined river for video stream \( j \), within which the “transmission trajectory” must be placed. Due to the jittery bit rate these banks are “smoothed” by applying a piecewise linear approximation. This is done by joining the points at both ends of an interval, by a linear interval, and then “gluing” the segments together.

Let \( L_j'(t) \) be such a piecewise linear approximation of \( L_j(t) \), with breakpoints at epochs: \( \tau = 0, \Delta t, 2\Delta t, \ldots, n\Delta t \) and \( L_j'(0) = 0 \). Let the \( i \)th interval correspond to the time interval: \( [(i-1)\Delta t \ (i\Delta t)] \) \( (i = 1, \ldots, n) \). Then, the breakpoints of \( L_j'(t) \), for the \( j \)th video stream, are represented by the sequence of values, \( (i, \ l'_j) \), \( i = 0, \ldots, n \). Similarly, construct a piecewise linear approximation of \( U_j(t) \) with the same intervals as in \( L_j'(t) \).

\[ U_j'(t) = L_j'(t) + b, \quad U_j(0) = b \]  

(1)

Here, we assume all \( m \) clients have the same buffer limits \( b \). The breakpoints of \( U_j'(t) \) are the sequence of points:

\( (i, u'_j), \quad i = 0, \ldots, n \)

where

\[ u'_j = l'_j + b \]  

(2)

3.1. Effective buffer

In order to result in lossless-starvation free playback \( L_j''(t) \), \( U_j'(t) \) can be shifted by amounts \( \delta_j^- \) and \( \delta_j^+ \), to obtain \( L_j''(t) \), \( U_j''(t) \), respectively, where

\[ L_j''(t) = L_j'(t) + \delta_j^- \]  

(3)
\[ U_j'(t) = U_j'(t) - \delta_j^- \]  
and  
\[ \delta_j^+ = \text{Max}\{U_j(v) - U_j'(v), \; v = 1, 2, \ldots, F\} \]
\[ \delta_j^- = \text{Max}\{L_j(v) - L_j'(v), \; v = 1, 2, \ldots, F\} \]

This effectively squeezes the smoothed riverbanks to new positions \( \{L_j'(t), U_j'(t)\} \), which represent the banks of the reduced river within which the transmission stream must lie. This operation reduces the buffer size by a small percentage; and is effected mostly by the interval width, as well as, stream variability. We have analyzed this operation and found it at worst to be \( 98\% \) of the original bank size. We refer to the width of this reduced river as the "effective buffer" for video stream \( j \).

It should be noted that \( L_j'(t) \) is the same as the \( L \) curve in the PCRTT scheme. Denote the "effective buffer", for client video stream \( j \), as
\[ b_j = U_j'(t) - L_j'(t) \quad \text{for all } t \]

Its relation to the actual buffer size, \( b \), is
\[ b_j = b - (\delta_j^+ + \delta_j^-) = b - \beta_j \]

where \( \beta_j \) is said to be the \( j \)th clients "wasted buffer size".

The breakpoints for interval \( (i+1) \) are shown in Fig. 1. A transmission schedule for the \( j \)th video stream is represented by the sequence
\[ S_j = \{y_{ij}, y_{i+1j}; \; i = 0, \ldots, n\} \]

Such a schedule is said to be a feasible if
\[ \{L_j'(i) \leq y_{ij} \leq U_j'(i), \; i = 0, \ldots, n\} \]

The dashed line in Fig. 1 shows a feasible transmission schedule. Let \( r_{ij} \) be the rate, over interval \( i \), of the \( j \)th video stream.
\[ r_{ij} = \frac{y_{ij} - y_{i-1j}}{\Delta t}, \quad i = 1, \ldots, n \]

where
\[ r_{ij} \geq 0, \quad i = 1, \ldots, n, \; j = 1, \ldots, m \]

Then the total rate, for all video streams over interval \( i \) is
\[ r_i = \sum_{j=1}^{m} r_{ij} \]

Heretofore, without loss of generality, set \( \Delta t = 1 \).
4. Linear programming formulation

In this section we first show that the optimal multiplexing problem can be formulated as a pure linear programming problem. We then perform a transformation on the transmission \((y_{ij})\) variables, so that their values are relative to the cumulative lower bound curves. In this improved formulation the variables become upper bounded. The following notation is helpful in describing the linearization process.

4.1. The optimal multiplexing problem

The Optimal Multiplexing Problem is to find the set of schedules \(\{S_j; j = 1,\ldots,m\}\), over all feasible sets in \(S\), such that the maximum rate (peak rate) is minimized, i.e.;

Find \(Z = \min \max \{r_i; i = 1,\ldots,n\}\) \((12)\)

This problem is in the form of a nonlinear mathematical program with linear constraints \((8)\)–\((11)\). To convert it into a pure linear programming problem, define a new non-negative variable \(R\) as the peak rate over all \(n\) intervals.

\(R = \max\{r_i; i = 1,\ldots,n\}\) \((13)\)

where

\(Z = \min R\) \((14)\)

Using \((10)\) and \((11)\) and the fact that \(R \geq r_i; i = 1,\ldots,n\) we obtain

\[ \sum_{j=1}^{m} r_{ij} \leq R, \quad i = 0,\ldots,n \] \((15)\)

\[ R \geq 0 \] \((16)\)

This eliminates the variables \(r_i\), so that the basic multiplexing problem (MP) is now:

**Problem MP** (linear programming formulation)

\(Z = \min\{R: R \geq 0\}\)

Subject to: \((8)\),\((9)\),\((10)\),\((15)\)

Constraint \((10)\) is necessary to insure that the rates are non-negative. The problem size is: \(n(m+1)\) linear constraints, \((n+1)m\) upper–lower bounded variables, and \(nm+1\) non-negative variables. Problem MP simultaneously solves the minimization peak total rate leveling problem, and the video transmission plans of the individual video streams. Thus, the solution provides the optimal rate schedules \(\{r_{ij}^0; i = 0,\ldots,n, j = 1,\ldots,m\}\). It also finds the minimum peak value, \(R^0\), of the combined rate function \(\{r_{ij}^0; i = 1,\ldots,n\}\); as well as, the optimal transmission schedules \(S_j^0 = \{y_{ij}, r_{ij}^0, y_{ij}^0; i = 0,\ldots,n\}\).

4.2. Conversion of MP to an upper bounded LP

A transformation of the \(y_{ij}\) variables is made which converts them to upper bounded variables \(y_{ij}^*\) by subtracting \(L_j^0(t)\) from \((8)\).

\[ l_{ij}^* \leq y_{ij}^* \leq u_{ij}^*, \quad i = 0,\ldots,n, \quad j = 1,\ldots,m \] \((17)\)

Since \(l_{ij}^* = 0\) and \(u_{ij}^* = U_j^0(i) - L_j^0(i)\), and by using \((5)\), the new variables become upper bounded of the form

\[ 0 \leq y_{ij}^* \leq b_j, \quad i = 0,\ldots,n, \quad j = 1,\ldots,m \] \((18)\)

The new upper bounded version of problem MP is now:

**Problem MP’** (upper bounded LP formulation)

\(Z = \min\{R: R \geq 0\}\)

Subject to:

\(\{y_{ij}^* - y_{ij}^0\} \leq (l_{ij} - l_{ij}^0), \quad i = 1,\ldots,n, \quad j = 1,\ldots,m \) \((19)\)

\[ \sum_{j=1}^{m} y_{ij}^* - \sum_{j=1}^{m} y_{ij}^0 + R \geq \left( \sum_{j=1}^{m} l_{ij} - \sum_{j=1}^{m} l_{ij}^0 \right), \quad i = 1,\ldots,n \] \((20)\)

\[ 0 \leq y_{ij}^* \leq b_j, \quad i = 0,\ldots,n, \quad j = 1,\ldots,m \] \((18)\)

Constraint \((19)\) insures that the rates \(r_{ij}\) do not become negative. Constraint \((20)\) insures that for each interval the value of the combined rate function does not exceed the peak rate \(R\). It is easy to see that Problems MP and MP’ are equivalent.

After obtaining the optimal solution to MP’, a simple post optimal calculation using, \(y_{ij}^* = y_{ij} - l_{ij}\) is required to recover the optimal rate schedules \(\{r_{ij}^0; i = 0,\ldots,n, j = 1,\ldots,m\}\), and the combined rate function \(\{r_{ij}^0; i = 1,\ldots,n\}\). The problem size of MP’ is the same as for MP except the for the fact that the \((n+1)m\) upper–lower bounded variables are replaced by upper bounded non-negative variables. This is important because it allows the use of the more efficient special upper bounded variable technique for linear programming. Moreover, this provides the advantage of facilitating a parametric analysis of the important factors of the problem.
such as buffer limits as they are placed on the right hand-side (RHS).

5. Multiplexing-buffer allocation problem

It is of interest to examine the effect of total buffer size on the optimal peak combined rate. To achieve this a total buffer size constraint may be added to Problem MP$. This also allows the use of the parametric analysis technique of linear programming to obtain the trade off between the optimal total rate and the buffer size (see Section 5.2).

5.1. Inclusion of buffer allocation

Let $B_T$ be the joint buffer size (a constant), and $b_j$ be the buffer size of the $j$th stream.

\[
\sum_{j=1}^{m} b_j = B_T \quad (21)
\]

\[ b_j \geq 0, \quad j = 1, \ldots, m \quad (22)\]

The solution will provide the optimal allocation of the total buffer to the buffers of all the video streams. In addition, it is possible to place upper limits on the individual buffer sizes if desired by adding constraints (23). Let $B_j$ equal a given upper bound on the $j$th stream buffer size

\[ b_j \leq B_j, \quad j = 1, \ldots, m \quad (23)\]

Define problem MP-B as the optimal multiplexing-buffer allocation problem.

Problem MP-B (multiplexing-buffer allocation problem)

\[ Z = \min \{R : R \geq 0\} \]

Subject to: (19),(20),(18),(21) and (22)

Of course, it is also possible to consider the case of identical individual buffer sizes. One simply solves MP-B where all $b_j$ in (23) are set to the same fixed value $B_j$. Two computational results for a sample problem of three video streams, and four time intervals are shown in Section 6, for case where $B_j$ equals 2 and 5 MB.

5.2. Parametric analysis—optimal total rate vs. buffer size

Tradeoffs between various parameters of the problem; and the optimal objective value can be generated using RHS ranging, a well-known technique in linear programming. For example, it may be of interest to examine the effect of total buffer size on the optimal peak rate $R$ in using problem MP-B. This can be achieved by performing RHS ranging of $B_T$ in constraint (21) in MP-B to obtain the function $\min Z(B) = R^c(B)$. It is known that for LP minimization problems, this function will be a non-increasing piecewise linear convex function of $B$. A numerical example with all $b_j$ in (23) set to $B_T/m$, resulted in such a function with the set of breakpoints: $[R^c(B_T),B_T] = [(14,0),(5,9),(0.5,27),(0,30)]$. The optimal solution also provides information on the shadow prices. Shadow prices, $\pi$, corresponding to (21), represent the marginal change in the optimal rate for one additional unit of $B_T$. These values, and the range of $B_T$ over which they are valid, are shown in Table 1. Of interest is the last column, which shows that if the buffer size is very large (over 30 in this case) increasing it by one unit has no effect on the optimal peak rate. One also sees from the curve, that very large buffer sizes result in the optimal peak rate of zero. In effect, we can pre-load the buffer at time zero and run it down without any transmission during playback. The decreasing values of $\pi$, as $B_T$ is increased, reflects the expected phenomenon of decreasing returns to scale.

6. Numerical example

Two small examples are solved with $n = 4, m = 3$, and buffer sizes fixed at $B_j = 5$ and 2, for each video stream $j$. The cumulative data breakpoints at time epochs $i$ are shown below. The optimal transmission schedules are shown in Table 2.

\[
L(j,i) = \begin{cases} 
0 & 1 & 4 & 5 & 10 \\
0 & 2 & 3 & 4 & 9 \\
0 & 4 & 5 & 6 & 10 
\end{cases}
\]

For $B_j = 5$, the total multiplexed schedule and rates are; [15, 18.5, 22, 25.5, 29] and [3.5, 3.5, 3.5, 3.5, 3.5], respectively. Note, that the multiplexed rate is flat over all intervals. For $B_j = 2$, the total multiplexed schedule and rates are; [0, 7, 15, 21, 29] and [7, 8, 6, 8], respectively. Here, the peak rate has been

<table>
<thead>
<tr>
<th>$B_T$</th>
<th>Shadow prices of buffer constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>$-1$</td>
</tr>
<tr>
<td>Range of $B_T$</td>
<td>0–9</td>
</tr>
</tbody>
</table>
increased to 8. For the $b_j$ equal 2 case, the results are plotted in Figs. 2 and 3.

7. Some results for real video streams

Here we present some results that were obtained from a simulation of real video streams. The experiments used 10 MPEG-1 video traces obtained from [2]. The video library includes clips with different lengths and subjects (high activity movies,

<table>
<thead>
<tr>
<th>$Y(j,i)$</th>
<th>Time epochs $i (B_j = 5)$</th>
<th>Time epochs $i (B_j = 2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5  5  8  10  10  0  1  6  7  10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5  5  5.5  5.5  9  0  2  4  6  9</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5  8.5  8.5  10  10  0  2  4  8  10</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. Optimum schedules (top: 3 individual streams, bottom: multiplexed stream).

Fig. 3. Zoom version of the optimal multiplexed stream.
MTV clips and news) that represent the diversity of compressed video sources in emerging multimedia services. Statistical characteristics of these movies are shown in Table 5. Each video stream was partitioned into 400 intervals each containing 35 frames. The algorithm was implemented in MATLAB®, and the LP model was solved using MOSEK® optimizer 3 [15].

The results of the multiplexed stream using the LP formulation (MP) was compared to those obtained by the ePCRTT and JSQ algorithms. We used ePCRTT due to its ease of computation in lieu of PCRTT. A comparison was carried out using several buffer sizes. Two cases for client buffer sizes of 0.6 and 10 Mbytes were analyzed. Fig. 4(a) and (b) shows a comparison of a ten multiplexed stream using ePCRTT, LP and JSQ for the 0.6 and 10 Mbytes cases, respectively. The server link bandwidth was limited to 10 Mbps. The dotted, solid and dash–dot curves represent LP, JSQ, and ePCRTT smoothing, respectively. Below we list the differences between the affects of applying each of the algorithms.

1. JSQ tends to use it maximum peak right at the beginning of transmission, due to its goal to fill all client buffers; after it reaches the status of all buffers full it behaves similar to ePCRTT. We see the effect of a larger buffer causes longer peak times in JSQ. This is reasonable, as it takes more start-up time to fill the client buffers.
2. ePCRTT is very diverse as expected, because of the jumpy movie rates; and its smoothing per interval only and not the whole as one.
3. LP provides the most constant server transmission plan, with the lowest peak rates. For the

![Fig. 4. Comparison between the multiplexed streams PCRTT, JSQ and LP for 0.6 and 10 MB buffer values.](image)

<table>
<thead>
<tr>
<th>Case (MB)</th>
<th>ORIG (unsmoothed) Mbps</th>
<th>PCRTT (smoothed) Mbps</th>
<th>LP (smoothed) Mbps</th>
<th>JSQ (smoothed) Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>8</td>
<td>8.232</td>
<td>5.3505</td>
<td>9.96</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>8.232</td>
<td>3.235</td>
<td>9.97</td>
</tr>
</tbody>
</table>
larger buffer size the LP solution is a perfect level rate, minimizing server-bandwidth variability. Table 3 provides a comparison of the peak rates. LP smoothing reduces the peak PRCTT rate 47% and 60% for the 0.6 and 10 Mbytes cases, respectively. When compared to JSQ the LP solution achieves reductions of 56% and 68%, for the 0.6 and 10 Mbytes cases, respectively. For comparison purposes, column 2 shows the peak rate required to transmit the unsmoothed original multiplexed stream. Note, that the LP reduces the unsmoothed peak rate by 46% and 60% for the 0.6 and 10 Mbytes cases, respectively.

Fig. 5 shows the effect of actual buffer size, b, on the peak-multiplexed rate using the LP procedure. The buffer quantity, b, is identical for each of the 10 streams and varied from 1 to 40 Mbytes. The server’s max rate can reach zero at high capacity buffer sizes. This is due to the phenomena that all the data can be pre-loaded before the first interval, which is equivalent to offline viewing.

Table 4 shows the wasted buffer sizes (βj) used to convert the actual buffer sizes to the effective buffer values used in constraint (18) of the LP formulation (for 1 MB buffer size, and 10 Mbps server bandwidth). The function is infeasible for values below \(b^* = 0.098\). At this point at least one client’s linear approximation of the river has an effective buffer size of zero. Note, that the ratio of the wasted buffer size from the given buffer size (1 MB in our case) is around the 0.1%. As expected from LP theory the function is a convex non-increasing piecewise linear function. One observes the decreasing shadow prices for the buffer as reflected by the slopes of the \(R^\circ(b)\) curve.

8. P-loss

Here we compare the ePCRTT, and LP algorithms to the JSQ procedure, by investigating the information loss due to exceeding the constant channel bandwidth by the multiplexed stream. The experiments are conducted for ten movie streams. The probability of loss is expressed by the expected fraction of bits lost from the video streams during transmission over a CBR channel. In general, to obtain an accurate approximation of \(P_{loss}\), it is essential to perform a simulation of the suggested multiplexed scheme during a large number of frame times. In this specific case, however, we want to compare the optimal LP scheme to the multiplexed stream obtained by ePCRTT. Thus, we compute the loss probability \(P_{loss}\) as follows:

\[
P_{loss} = \lim_{n \to \infty} \frac{\text{number of bits lost up to } n}{\text{total number of bits sent up to } n} \tag{30}
\]

\[
P_{loss} = \lim_{T \to \infty} \frac{\sum_{i=1}^{T} |r_i^o - C|^+}{\sum_{i=1}^{T} r_i^o} \tag{31}
\]

where

\[
|x-y|_+ = x-y, \text{ if } x-y > 0, \text{ and } 0, \text{ otherwise.}
\]

Here \(C\) is the channel rate and \(r_i^o\) is the optimal multiplexed rate during time interval \(i\). Eq. (31) represents the accumulative number of bits above the channel bit rate divided by the total number of bits in the multiplexed stream. The \(P_{loss}\) is obtained as function of the channel bit rate, \(C\), for decreasing \(C\) from the maximum value of the multiplexed stream until a zero bit rate is reached. \(P_{loss}\) is a function decreasing in \(C\). When the rate is perfect

![Fig. 5. Peak rate, \(R^\circ\), vs. real buffer size, \(b\).](image)

<table>
<thead>
<tr>
<th>Video (j)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>(b^*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\beta_j) (Mbytes)</td>
<td>0.0597</td>
<td>0.0347</td>
<td>0.0551</td>
<td>0.0691</td>
<td>0.0322</td>
<td>0.1023</td>
<td>0.0297</td>
<td>0.0312</td>
<td>0.0783</td>
<td>0.0522</td>
<td>0.1023</td>
</tr>
</tbody>
</table>
(a constant peak rate) $P_{\text{loss}}$ function is linear decreasing to zero (when $C = \text{peak rate}$). The $P_{\text{loss}}$ function represents unevenness by a piecewise linear concave decreasing curve (for incremental rate intervals). The graphs in Fig. 6(a) and (b) show log scaled $P_{\text{loss}}$ as a function of the channel bit rate.

Fig. 6. $P$-loss vs. channel bandwidth for PCRTT, JSQ and LP for buffer sizes of 0.8 and 10 Mbytes.
for client buffer size cases (all clients have identical buffer sizes) of 0.8 and 10 Mbytes, respectively. Curves of $P_{\text{loss}}$ for ePCRTT, LP and JSQ are plotted. As expected, the $P_{\text{loss}}$ is decreasing, for all cases, as a function of the channel rate.

We conclude from both plots that LP method requires a minimum of Server link bandwidth (5 and 2.8 Mbps) in both plots compared to JSQ and ePCRTT. Performance of the LP scheme increases as the buffer size increases, and the gap between the three $P_{\text{loss}}$ curves is more noticeable for larger buffer sizes. This is because, for a large buffer, there are more possible solutions for the LP, and the range of rates in each smoothing interval is larger. In the ePCRTT case, enlargement of the buffer size has no influence on the packet-loss curve. However, the JSQ behavior as increasing the buffer size is different. The worst case for JSQ was accepted for a given buffer size of 10 MB, while for the case of 0.8 MB buffer size the performance of the JSQ are reasonable. The explanation for this result is that the JSQ tries to fill all the clients’ buffers, and to balance buffer usage between all clients. Hence, larger buffer sizes can degrade JSQ performance, because in this case the algorithm tries to fill many empty buffers simultaneously.

9. Effect of number of movies and interval approximation

Fig. 7 plots two measures for average variability within each stream of the JSQ and LP multiplexing schemes, as a function of the number of movies. The underlying environment for this experiment is: 10,000 frames for each movie, buffer size of 5 Mbytes, and a server bandwidth of 7 Mbps. LP peak rate increases as more movies are added. This is because there is more data to stream, and more buffers space to utilize for this purpose. The JSQ peak rate is almost on a constant at a rate of 7 Mbps, which is due to the fact that JSQ strives to fill buffers, which always are in demand. The LP peak rate is lower than that of the JSQ even for the largest number of movies used. The standard deviation in the LP scheme is approximately zero, representing a stabilized bandwidth allocation during the 10,000 frame time interval. The JSQ standard deviation increases until five movies are reached, and from that point decreases to 0 at 13 movies. As more movies are added, JSQ reaches a point where it distributes the server buffer almost evenly to all clients, which results in a more stable inner variability. Until 13 movies are included there

![Fig. 7. Peak rate and standard deviation of server bandwidth, for JSQ and LP, vs. number of movies.](image-url)
is enough bandwidth, so that it will reach full buffer capacity. Hence the jumping between streams is more related to the movie frames variability (Table 5). Beyond the 13 movies point, the objective of JSQ is to equalize the buffer usage; and since it cannot achieve full buffer usage the result is in a more equalized bandwidth distribution (hence the zero variability).

In addition the effect of increasing the number of intervals in the piece linear approximation of the accumulated frame data in the video stream was studied. Fig. 8 illustrates the results by plotting the peak rate and standard deviation vs. number of intervals. It can be seen that for the LP approximation as the number of intervals are increased the peak rate increases approximately linearly.

10. Conclusions and future research

In this paper we proffer a linear programming formulation of the optimal multiplexing problem. Although there are several other approaches for solving this problem the advantage of the present approach are many. Linear programming is a well-known algorithm, easily understood, and readily available without the need for special coding. Our formulation accepts a linear approximation to the problem by partitioning the video streams into equal periods of length $\Delta t$. This approximation can be made more exact by decreasing $\Delta t$ as desired. It will also accept an initial approximate solution to the program for further speedup. It is flexible, allowing the addition of constraints without the necessity of algorithm redesign. For example, if one wishes to add a bandwidth limit for each client, this can easily be done by placing an upper bound on the rate variables, $r_{ij}$. Sensitivity analysis is another

<table>
<thead>
<tr>
<th>Movie</th>
<th>Average frame size (bits)</th>
<th>Standard deviation</th>
<th>Peak-mean ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asterix</td>
<td>22,600</td>
<td>2.0218</td>
<td>5.4917</td>
</tr>
<tr>
<td>Bond</td>
<td>17,845</td>
<td>1.8627</td>
<td>8.3089</td>
</tr>
<tr>
<td>Dino</td>
<td>12,889</td>
<td>1.3968</td>
<td>8.3215</td>
</tr>
<tr>
<td>Fuss</td>
<td>27,262</td>
<td>2.5721</td>
<td>6.6903</td>
</tr>
<tr>
<td>MrBean</td>
<td>15,950</td>
<td>1.8812</td>
<td>14.361</td>
</tr>
<tr>
<td>Mtv1</td>
<td>25,512</td>
<td>2.2744</td>
<td>8.9840</td>
</tr>
<tr>
<td>Mtv2</td>
<td>19,120</td>
<td>1.7323</td>
<td>10.432</td>
</tr>
<tr>
<td>News1</td>
<td>19,602</td>
<td>2.5685</td>
<td>9.9182</td>
</tr>
<tr>
<td>Race</td>
<td>31,074</td>
<td>2.2489</td>
<td>5.9873</td>
</tr>
<tr>
<td>Simpsons</td>
<td>21,125</td>
<td>2.1277</td>
<td>6.7859</td>
</tr>
</tbody>
</table>

Fig. 8. Peak rate and standard deviation vs. number of intervals.
advantage. For example, the shadow price of any individual buffer limit represents the marginal change of the optimal rate for an additional increment of buffer size. It also allows the investigation of varying selected parameters of the problem over their entire range through the use of parametric programming. This is demonstrated for the total buffer size parameter for both a small numerical example and for real video streams. It also, provides a framework for the future inclusion of multiple objectives, thus, tapping the power of multi-objective programming techniques. Finally, linear programming problems have been proven to belong to the class of polynomial bounded problems.

The optimal LP multiplexing scheme has been applied to real MPEG-1 video sequences. These results are compared to the ePCRTT and JSQ procedure in terms of peak transmission bandwidth, ratio and $P_{loss}$ performance measures. For several client buffer sizes, the rate obtained by ePCRTT was reduced by up to 60% by the LP solution, and compared to JSQ the LP procedure found peak rates by up to 68% lower. Future work includes consideration of the dynamic VoD environment, and a study of secondary measures and techniques for further smoothing of the rate function. In addition, we intend to investigate the effect of playback delay, and buffer size modifications to account for errors induced by the linear approximation and real-time usage of the LP method.

Acknowledgments

This research was partially supported by the Paul Ivanier Center for Robotics Research & Production Management, Ben-Gurion University of the Negev. Special thanks are given to Niv Caner and Eina Titkin for their help in coding and running portions of the simulation software.

References


Further reading