Bandwidth-Efficient Continuous Media Streaming Through Optimal Multiplexing

Wei Zhao
Mobile Computing and Multimedia Lab
Department of Computer Science
University of Maryland
College Park, MD 20742
zw@cs.umd.edu

Satish K. Tripathi
Bourns College of Engineering
University of California
Riverside, CA 92521-0425
tripathi@engr.ucr.edu

Abstract

Maximizing bandwidth efficiency in distributed continuous media streaming systems is the key in delivering cost-effective multimedia services to distributed and heterogeneous receivers. We introduce a technique based on stream multiplexing to achieve the highest possible bandwidth efficiency, while preserving stringent and deterministic quality of service guarantees. The technique accomplishes the optimal multiplexing (i.e. resulting in the lowest possible bandwidth allocation) by exploiting both the temporal and the spatial structures among a group of continuous media streams. We present a family of optimal multiplexing schedules. The adverse per-stream effects of optimal multiplexing are studied and a technique based on transmission rearrangement is proposed to mitigates these effects, without sacrificing the achieved multiplexing optimality. The results presented in the paper provide some fundamental criteria and limits in the design and evaluation of resource allocation, admission control and stream scheduling policies for bandwidth efficient continuous media streaming.

Keywords: Multimedia Streaming, Transmission Scheduling, Multiplexing, Bandwidth Allocation, Admission Control, Temporal Smoothing, Feasible Region, Quality-of-Service

1 Introduction

The advance of network technology towards high bandwidth and ubiquitous connectivity (e.g. Intranet, Internet, Wireless, Satellite, Cellular, etc.), as well as the progressively more diversified, more powerful and cheaper end-user devices (e.g. PCs, TV set-top boxes, wireless hand-held devices, etc.) are establishing a solid infrastructure upon which the proliferation of multimedia communications is envisioned in the near future. Although new technologies developed in the recent years allow most continuous media (e.g. video and audio) to be encoded, stored and transported in compressed forms that require much lower data rates, the delivery of continuous media with decent quality from a server over the network is still bandwidth intensive. To provide cost-effective continuous media streaming services in environments with finite bandwidth resources, improving the bandwidth efficiency is the key.

One of the major factors affecting the bandwidth efficiency of streaming is the high variability in data rate over the durations of continuous media streams [6]. The high variability is contributed by both the inherent media content dynamics and the use of prediction-based encoding techniques (e.g. inter-frame coding in MPEG video [10]) for higher compression performances.

Traditionally, peak-rate allocation is used to allocate network bandwidth for the streaming of variable-bit-rate (VBR) continuous media streams. However, peak-rate allocation can be quite wasteful of network resource if the bit-rate variation is high. Although there are other network services such as ATM VBR service aimed at transporting VBR sources, due to their statistical nature, they are not suitable for the transport of continuous media with deterministic quality-of-service (QoS) guarantees.

Temporal smoothing techniques [15, 5] have been proposed over the past few years. The techniques take a work-ahead approach by delivering and buffering media data some time prior to their playback. By carefully scheduling the transmission of media data (e.g. video frames) over time, the receiver is able to playback the continuous media stream without playback starvation or buffer overflow. When the media stream profile (e.g. frames sizes) is known *a priori*, an optimal temporal transmission schedule can be calculated that achieves the lowest possible bandwidth requirement. Temporal smoothing has been shown to be very effective [15] in reducing the bandwidth allocation of individual continuous media streams.

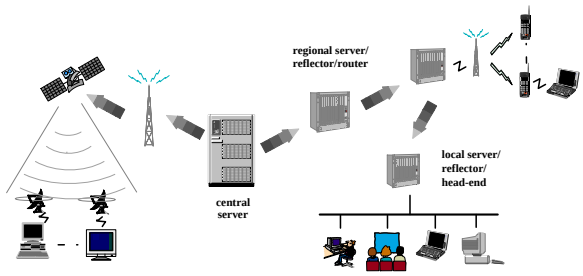


Figure 1: Typical media distribution scenario

In typical media-on-demand scenarios (Figure 1), continuous media are being streamed from a media server to a set of distributed receivers through a combined end-system and network distribution tree. Portions of the distribution tree,

including the media storage device, the network interface adapter at the media server and the shared network path, are being shared by groups of on-going continuous media streams. With multiple VBR streams sharing a common resource, there is the potential of reducing the overall allocated bandwidth (i.e. disk, interface or network bandwidth) at the shared resource by inter-stream multiplexing. In a few recent works [8, 17], phase-based multiplexing techniques are proposed to achieve higher bandwidth efficiencies for MPEG video streams. Taking advantage of the Group-Of-Picture (GOP) frame structure of MPEG [10], periodic phase-based time-varying traffic envelopes are used to characterize the bit-rate of the streams, based on which multiplexing schedules are derived with the purpose of reducing the overall bandwidth allocation.

In this paper, we propose techniques exploiting the advantages of both temporal smoothing and spatial multiplexing. Unlike previous studies involving statistical multiplexing (e.g.[16]) where statistical or probabilistic guarantees were provided, the techniques proposed in this paper achieve higher bandwidth efficiency while maintaining stringent and deterministic quality of service guarantees. In fact, we show that the proposed technique achieves the *optimal multiplexing* in the sense that it requires the least possible bandwidth among all valid streaming schedules.

There have been studies on stream-sharing techniques to improve bandwidth efficiency, where a single stream originated from the server is shared by multiple requests or receivers. In "stream-batching" [2, 1], requests for the same continuous media stream are batched and served with a single stream, if the requests are within a certain time interval of each other. Bandwidth is saved by allocating a single stream at the server for multiple requests, but at the cost of additional startup delays due to batching. In an alternative technique called "adaptive piggybacking" [7], streams are started without additional waiting time, but are played-back at slightly different speeds. As playback progresses, streams will eventually "merge into" other on-going streams to save bandwidth.

Stream-sharing techniques are orthogonal and mutually-complimentary to the stream multiplexing techniques proposed in this paper. Stream multiplexing can improve the bandwidth efficiency for the delivery of a set of concurrent media streams that can not be further trimmed down using stream-sharing. Stream multiplexing can also work with other streaming techniques, such as adaptive streaming [12, 13] in environments with fluctuating resource conditions, to improve their bandwidth efficiencies.

We present a family of optimal multiplexing schedules that achieve the bandwidth lower bound. The multiplexing schedule is then mapped into individual per-stream schedules that can be executed by the server. However, optimizing the multiplexing bandwidth is often realized at the cost of suboptimal per-stream bandwidth efficiencies. In some cases, the resulting per-stream bandwidth becomes so high that it offsets any achieved multiplexing bandwidth savings. To overcome this deficiency, we propose a revised per-stream scheduling technique to mitigate the adverse per-stream effects while still maintaining the same optimal multiplexing bandwidth.

The rest of this paper is organized as follows. In Section 2, we describe the continuous media streaming process and its fundamental resource requirements. In Section 3, we introduce the basic notions and properties of stream multiplexing. In Section 4, we establish the exact bandwidth lower bound for the streaming of a set of continuous media streams and present a family of optimal multiplexing schedules. Transmission rearrangement of per-stream schedules is introduced in Section 5. Numerical results based on real compressed video traces are shown in Section 6.

2 Continuous Media Streaming and Resource Requirement

Figure 2 shows a typical multimedia streaming system. Multimedia contents are being retrieved from a media server (or a network of servers) where the contents are stored, and streamed over a network distribution tree, to a set of distributed, possibly heterogeneous receiving devices where the content are being rendered. We use the term *streaming* to distinguish itself from the alternate approach of downloading and playing-back. Streaming has clear advantages over downloading: it provides a much smaller playback startup delay and requires a much smaller storage buffer to temporarily store the received multimedia content.

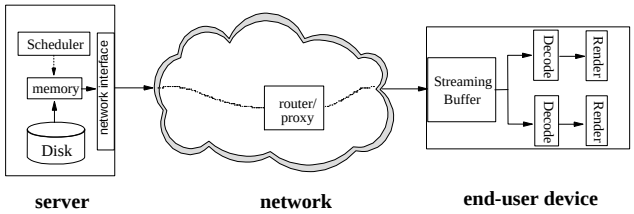


Figure 2: Components of a continuous media streaming system

2.1 System Components and Scheduling Constraints

Figure 2 shows the components of a continuous media streaming process from a server to a receiver. The end-to-end streaming path consists of components and resources from the server, the network and the receiver. After a streaming request is received and processed by the server, the requested continuous media is retrieved from the server's storage unit, packetized and sent through its network interface adapter, over a network path to arrive at the receiver's network interface. When the receiver receives a data packet, it stores the received data temporarily in its *streaming buffer*. A rendering thread running on the receiver periodically picks up the media elements (e.g. video frames) from the streaming buffer, executes the decoding routine and renders the media elements (screen display or audio playback) at their scheduled presentation times. After a media element is picked up from the streaming buffer, it is no longer useful and is discarded to free up its buffer space for incoming media traffic.

Let the continuous stream consist of N media elements, each media element o_i having a data size of $s(i)$ and a deadline of $t(i)$ representing its scheduled playback time. There is a streaming buffer of size M at the receiver side to temporarily store the incoming stream traffic. The process can be represented by a *feasible region* in Figure 3. The horizontal axis represents time t and the vertical axis represents the cumulative amount of data y received by the client. A transmission schedule is then represented by a *monotonically nondecreasing* curve starting at $y = 0$. The slope of the transmission curve at any point is therefore the instantaneous transmission rate at that time. To avoid starvation of the rendering thread, media elements must arrive before their respective deadlines. Hence a *lower bound curve* $L(t)$ is defined to represent the minimum cumulative

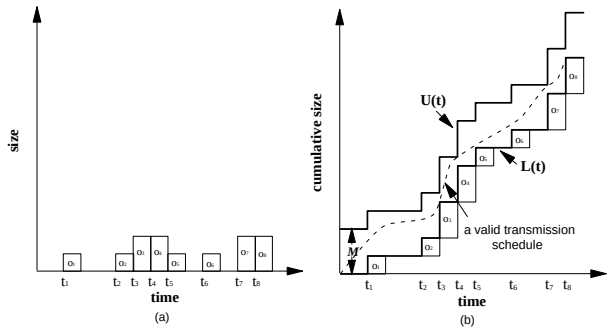


Figure 3: A continuous media stream and its feasible region

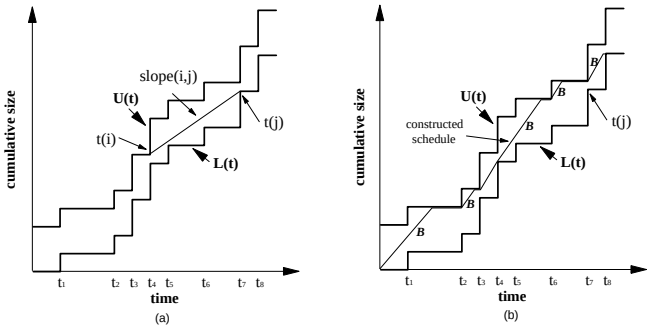


Figure 4: Slopes from $t(i)$ to $t(j)$ and constructed schedule with peak rate B

amount of data that should have been received by time t, $L(t) = \sum_{t(i)\leq t} s(i)$. On the other hand, the streaming buffer has only a finite size of M. To avoid buffer overflow, an *upper bound curve* $U(t)$ is defined to represent the maximum amount of data that can be delivered by time t. Thus $U(t)$ lies M units above $L(t)$ immediately *after* a stepping point of $L(t)$, when the buffer space for the due media element is freed up, $U(t) = L(t^-) + M$ for $t > 0$, and $U(0) = 0$. For convenience, let $t(0) = 0$. Therefore, any part of a schedule curve lying below $L(t)$ leads to playback starvation, while any part lying above $U(t)$ leads to buffer overflow. $L(t)$ and $U(t)$ define the boundaries of a feasible region. A transmission schedule is valid if and only if the transmission curve lies within the feasible region.

2.2 Fundamental Bandwidth Requirement

Among all valid transmission schedules, we are most interested in the ones that are most bandwidth efficient (i.e. requiring the least amount of bandwidth). Bandwidth efficient schedules maximize the utilization of the available bandwidth (with or without reservations) and are more likely to be admitted successfully in a reservation-based system. The following lemma establishes the exact bandwidth lower bound of a continuous media streaming process.

Lemma 1 *Let B^* be the exact lower bound on the peak rate of any valid transmission schedules, $B^* = \inf\{Peak(S) | S \text{ is valid}\}$, where $Peak(S)$ is the peak rate of schedule S. Then,*

$$B^* = \max\{\max_{0\leq i<j\leq N}(L(t(j)) - U(t(i)))/(t(j) - t(i)), 0\}$$

Proof: Let B represent the right side of the equation. We first show that $B^* \geq B$. To show this, consider the streaming process from time $t(i)$ to $t(j)$ (Figure 4(a)). any valid schedule S must be at or below $U(t(i))$ at time $t(i)$ and be at or above $L(t(j))$ at time $t(j)$. Therefore, the peak rate of S between $t(i)$ and $t(j)$ must be at least the slope of the segment linking $(t(i), U(t(i)))$ to $(t(j), L(t(j)))$, which is $(L(t(j)) - U(t(i)))/(t(j) - t(i))$. By the facts that this holds for any $0 \leq i \leq j \leq N$ for any schedule S and that the schedule slopes are always non-negative, $B^* \geq B$ holds.

We then show $B^* \leq B$ by constructing a valid schedule S with peak rate B from $(0, U(0) = 0)$. S maintains slope B until it reaches U, at which point it takes slope 0 until it reaches the closest stepping point of U where it can resumes slope B. The process repeats till the end of the stream (Figure 4(b)). Obviously S has a peak rate B. S is also valid since the segments with slope B originating from U never penetrate L first. Otherwise, we would have found a segment linking U to L with a higher slope than B, contradicting B's definition.

□

2.3 The Inter-frame and the Optimal Transmission Schedules

A valid schedule can be easily derived by sending each media element at a constant rate in the interval between the previous deadline and its deadline. We call the schedule derived this way the *inter-frame schedule*, as shown in Figure 5(a).

An optimal temporal transmission scheduling algorithm was presented in [15]. The algorithm produces a piecewise linear transmission schedule that is as smooth as possible, with the minimum peak rate and rate variance over all valid transmission schedules. The algorithm is straightforward: starting with $t = 0$, the algorithm looks for the longest segment within the feasible region (Figure 5(b)). If the segment ends on the lower bound curve, it generates a schedule segment ending at the latest upper bound it touches, and vice versa. The procedure is then repeated from the new start point. The algorithm is essentially identical to the shortest path algorithm in the feasible region [9]. An equivalent algorithm with running time $O(N)$ was presented in [14].

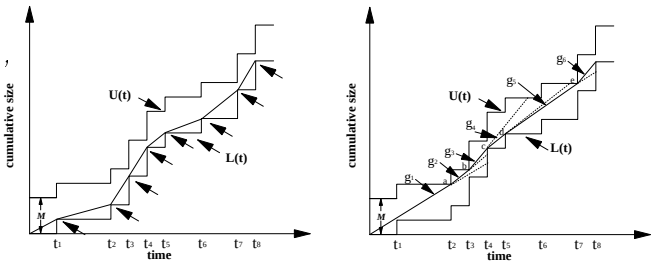


Figure 5: The inter-frame and the optimal transmission schedules

3 Fundamentals on Stream Multiplexing

3.1 Stream Aggregations

We first consider a simplified scenario of stream multiplexing where all streams are being delivered to the *same* destination with a *shared* streaming buffer. The size of the shared streaming buffer is thus the sum of the individual streaming buffers. In such scenario, we can actually aggregate the multiple continuous media streams and treat them equivalently as a single stream. We define this as *stream aggregation*,

an example of which is shown in Figure 6. Essentially, a stream aggregation is a stream consisting of all media elements (with their respective deadlines and sizes unchanged) from all the component streams, being delivered to a receiver with a combined streaming buffer. We represent the streaming aggregation consisting of streams $s_1, s_2, \ldots, s_K$ as $\sum_{i=1}^{K} s_i$, where K is the number of continuous media streams.

Lemma 2 *The feasible region of the stream aggregation is* $(\sum_{i=1}^{K} L_i, \sum_{i=1}^{K} U_i)$*, where* (L_i, U_i) *is the feasible region of component stream* s_i*.*

Proof: By definition, the lower bound curve of the aggregation feasible region at time t is $L(t) = \sum_{t(i) \leq t} s(i)$, where $s(i)$ and $t(i)$ are the size and deadline of media element o_i respectively. Since the stream aggregation contains all the media elements from all the component streams, the above is equal to $\sum_{j=1}^{K} \left(\sum_{t(i) \leq t, o_i \in s_j} s(i) \right) = \sum_{j=1}^{K} L_j(t)$. As for the upper bound curve, since $U(t) = L(t^-) + M$, where $M = \sum_{i=1}^{K} M_i$ is the aggregation's streaming buffer size and M_i is the streaming buffer size of s_i. Therefore, $U(t) = \sum_{i=1}^{K} L_i(t^-) + \sum_{i=1}^{K} M_i = \sum_{i=1}^{K} \left(L_i(t^-) + M_i \right) = \sum_{i=1}^{K} U_i(t)$. □

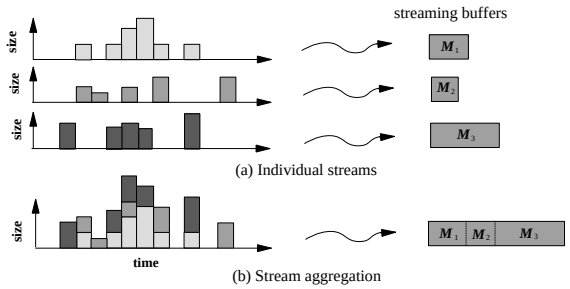


Figure 6: Individual streams and stream aggregation

In other words, the feasible region of the stream aggregation is easily obtained by summing up the lower and upper bound curves of the individual feasible regions. We can run the optimal schedule algorithm on the aggregated feasible region. The resulted schedule is the optimal schedule for the transmission of the stream aggregation. We call this schedule the *optimal aggregation schedule*, represented by S^*_Φ for the stream set $\Phi = s_1, s_2, \ldots, s_K$.

3.2 Bounding Multiplexing Gain

We turn back to the original setting of streaming continuous media to distributed, possibly heterogeneous receivers. Compared to stream aggregation, one crucial difference is the nature of interactions between individual streams and the individual streaming buffers. In the stream aggregation scenario, the streaming buffer is fully shared by all component streams. But in distributed streaming, each receiver has its own streaming buffer. It is impossible to share the streaming buffers among different streams because of the distributed nature of the receivers.

When the streaming buffer is fully shared, each stream can take buffer space up to the entire aggregated streaming buffer to temporarily store its delivered media data. Compared to distributed streaming where each stream can only store data up to its own streaming buffer, streaming with a shared buffer has more scheduling flexibility, thus may result in higher bandwidth reduction through multiplexing. In the following, we study how the isolated streaming buffers affect stream multiplexing. We start with a few results establishing the multiplexing lower bound.

Let S_i be a valid schedule of stream s_i, we define the sum of the schedules $\sum_{i=1}^{K} S_i$ as the sums of the schedule curve functions $S_i(t)$. We observe the following property.

Lemma 3 *Let* S_i *be a valid schedule of stream* s_i*, then the sum of the individual schedules* $\sum_{i=1}^{K} S_i$ *is a valid schedule of the stream aggregation* $\sum_{i=1}^{K} s_i$*.*

Proof: Since S_i be a valid schedule of s_i, $L_i(t) \leq S(t) \leq U_i(t)$, where (L_i, U_i) is the feasible region of stream s_i. Summing up the inequalities over all i, $\sum_{i=1}^{K} L_i(t) \leq \sum_{i=1}^{K} S_i(t) \leq \sum_{i=1}^{K} U_i(t)$. By *Lemma 2*, $(\sum_{i=1}^{K} L_i, \sum_{i=1}^{K} U_i)$ is the feasible region of the stream aggregation $\sum_{i=1}^{K} s_i$. Therefore $\sum_{i=1}^{K} S_i$ is a valid schedule of $\sum_{i=1}^{K} s_i$. □

Let S_M be a valid multiplexing schedule consisting of individual schedules S_i for stream s_i, $S_M = \sum_{i=1}^{K} S_i$. S_i must be a valid schedule for stream s_i in order for the multiplexing schedule S_M to be valid.

Lemma 4 *Let* S_M *be a valid multiplexing schedule,* $Peak(S_M) \geq Peak(S^*_\Phi)$*, where* S^*_Φ *is the optimal aggregation schedule for the stream set* $\Phi = s_1, s_2, \ldots, s_K$*.*

Proof: Decompose S_M into valid individual schedules $S_i, i = 1, 2, \ldots, K$. By *Lemma 3*, $S_M = \sum_{i=1}^{K} S_i$ is a valid schedule of the stream aggregation $\sum_{i=1}^{K} s_i$. But S^*_Φ is the optimal schedule of the stream aggregation, hence $Peak(S_M) \geq Peak(S^*_\Phi)$ by the optimality of S^*_Φ. □

Therefore, stream aggregation represents a lower bound on stream multiplexing in terms of bandwidth requirement. Any bandwidth gain that can be achieved by stream multiplexing can be matched by stream aggregation. By *Lemma 2*, we have the feasible region of stream aggregation, from which we can calculate the optimal aggregation schedule S^*_Φ. The bandwidth achieved by S^*_Φ is a lower bound on the bandwidth achievable by multiplexing, no matter what multiplexing scheme is used.

3.3 Summed Optimal Schedule and Summed Inter-frame Schedule

A valid multiplexing schedule can be obtained by simply following arbitrary individual stream schedules as long as they are valid. Different multiplexing schedules are thus derivable using different individual schedules. The multiplexing schedule derived this way is simply a sum of the individual schedules. We define the *summed optimal schedule* as the sum of the optimal individual schedules and the *summed inter-frame schedule* as the sum of the inter-frame individual schedules which were described earlier. We use these two simple multiplexing schedules in our numerical comparisons later in the paper.

4 Optimal Stream Multiplexing

4.1 The Inevitable Multiplexing Bandwidth

Before deriving any bandwidth bound on stream multiplexing, we look at each stream individually and examine their

resource requirements separately. We turn to *Section 2* and examine Figure 4 of *Lemma 1*. The maximum amount of data that could have be delivered for a stream s_k by time $t(i)$ is upper bounded by $U_k(t(i))$ to avoid streaming buffer overflow; the minimum amount of data that should have been delivered for stream s_k by time $t(j)$ is lower bounded by $L_k(t(j))$ to avoid playback starvation. Thus the minimal amount of data belonging to stream s_k that needs to be delivered during the time interval $[t(i), t(j)]$ is $L(t(j)) - U(t(i))$, or zero if this value is negative (delivered data can not be retracted). Since this is true for every stream, the total amount of media data that needs to be delivered during the time interval $[t(i), t(j)]$ is at least $\sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}$. The peak bandwidth needed in this interval to deliver this amount of data is thus $\sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}/(t(j) - t(i))$. Taking the maximum of this expression over all possible i,j, we have the *inevitable multiplexing bandwidth* (i.e. bandwidth that is absolutely necessary) for the transport of a group of continuous media streams.

Lemma 5 *The inevitable multiplexing bandwidth to deliver a set of streams is* $max_{0 \le i < j \le N} \sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}/(t(j) - t(i))$.

Proof: Proof given by the above derivations. □

4.2 Eligible Earliest Deadline First Multiplexing

We introduce a family of multiplexing schedules called *Eligible Earliest Deadline First* (E-EDF) multiplexing. The schedule is based on the traditional *Earliest Deadline First* (EDF) task scheduling [11] that have been well-studied and widely applied in a number of different areas. We extend EDF to schedule the multiplexing and transmission of multiple continuous media streams.

EDF is a straightforward scheduling scheme, where tasks or objects are scheduled or processed in the order of their deadlines. The one with the earliest deadline is handled first. Despite its simplicity, EDF is an optimal scheduling scheme in the sense that it can duplicate any valid schedules [11] with the same amount of resource.

E-EDF extends EDF by imposing an eligibility constraint to each candidate media element. Only media elements that are *eligible* can be scheduled for transmission. A media element is defined as eligible if its transmission does not result in streaming buffer overflow at the receiver. Among all media elements that are eligible, E-EDF selects the one with the earliest deadline for transmission. Note that media elements can be partially eligible (when the residual buffer space does not hold the entire media element) and thus transmitted in portions, and that the tie-breaking rule among eligible media elements with the same deadline is unspecified, resulting in a family of schedules.

Definition 1 *S_M is an E-EDF multiplexing schedule if and only if the followings hold: at any time t when a media element o with deadline d is under transmission, o is eligible and for any other object p with deadline e not yet finished transmission, p is either ineligible or $e \ge d$. A media element from stream s_i is defined as eligible at time t if and only if the residual streaming buffer size $U_i(t) - S_i(t) > 0$.*

Despite its simplicity, we can show that E-EDF is an optimal multiplexing scheme. A multiplexing scheme is optimal if no other multiplexing scheme can produce a multiplexing schedule having a lower resource (in this case, bandwidth) requirement. We establish this by showing that it is impossible for any multiplexing schedule to be valid if an E-EDF schedule with the same bandwidth requirement is not valid. In other words, no other multiplexing scheme can possibly do better than E-EDF.

Theorem 1 *E-EDF multiplexing is optimal.*

Proof: Let S_M^{eedf} be an E-EDF multiplexing schedule consisting of per-stream schedules S_i^{eedf} for stream s_i, $i = 1, 2, \ldots, K$. Let S_M be an arbitrary multiplexing schedule with the same schedule curve (i.e. $S_M(t) = S_M^{eedf}(t), \forall t$, thus having the same bandwidth requirement) consisting of individual per-stream schedules $S_i, i = 1, 2, \ldots, K$. Suppose S_M is valid while S_M^{eedf} is not. We show this results in contradiction.

To trace the schedules as they progress over time, we divide the time-axis into small time-slices. Time-slices are delimited by critical events. There are two types of critical events: the arrival of a deadline and the start of the transmission of a new media element. We show at the end of each time-slice, the following claim is true: for every deadline d, the total amount of media data with deadline d or earlier transmitted in S_M^{eedf} is no less than that in S_M. The claim is shown by induction on the time-slices.

Initially, the claim holds trivially since no media elements have been transmitted yet. Suppose the claim holds at the end of time-slice l, we show it still holds at the end of time-slice $l+1$.

In S_M^{eedf}, the media elements (can be one or multiple) under transmission during time-slice $l+1$ must be eligible and have the same earliest deadline d' among all eligible media elements. Note that they consume all the bandwidth of S_M^{eedf}, which is also the bandwidth of S_M during time-slice $l+1$. Thus for any deadline $d \ge d'$, the entire bandwidth in time-slice $l+1$ is dedicated to the transmission of media elements with deadline d or earlier. By induction hypothesis, the claim holds for $d \ge d'$ at the end of time-slice $l+1$. For any deadline $d < d'$, the reason why media elements with deadline d' are being transmitted in S_M^{eedf} is that all media elements from other streams with deadlines less than d' not yet transmitted are all ineligible. In other words, their streaming buffers have been filled in S_M^{eedf} and the transmission of their media elements has reached their maximums by the end of time-slice l. For those streams with non-full streaming buffers, their media elements with deadlines $d < d'$ or earlier have already been totally transmitted in S_M^{eedf} by the end of time-slice l. So the amount of media elements with deadlines $d < d'$ or earlier that can be transmitted by the end of time-slice $l+1$ have already reached its upper bound in S_M^{eedf} by time-slice l. There is no way to achieve a higher amount in S_M by the end of time-slice $l+1$.

By induction, the claim holds. Now at any deadline d, the transmitted media element with deadline d or earlier in S_M^{eedf} is no less than that of S_M, contradicting the assumption that S_M^{eedf} leads to playback starvation while S_M does not.

□

4.3 Achieving the Inevitable Bandwidth

In this part, we show that the inevitable multiplexing bandwidth given in *Lemma 5* can actually be achieved by a fam-

ily of E-EDF multiplexing schedules. Consequently, the inevitable bandwidth is the optimal multiplexing bandwidth, achievable by a family of optimal E-EDF multiplexing schedules.

Before proceeding, we introduce a family of E-EDF multiplexing schedules called *Greedy E-EDF* multiplexing schedules. Given a certain bandwidth bound B, greedy E-EDF schedules use the entire available bandwidth B as long as there are eligible media elements to transmit, while following the earliest eligible requirement of E-EDF. Obviously, greedy E-EDF schedules under bandwidth B has a peak bandwidth requirement of B.

Definition 2 *A greedy E-EDF multiplexing schedule S_M^{geedf} under bandwidth B is an E-EDF multiplexing schedule with the following property: at any time t, if there are eligible media elements to be transmitted, $S_M^{geedf\prime}(t) = B$; otherwise, $S_M^{geedf\prime}(t) = 0$. Note that the derivative $S'(t)$ is the transmission rate of schedule S at time t.*

We establish the optimality of the inevitable bandwidth in *Lemma 5* by showing that a greedy E-EDF multiplexing schedule under the inevitable bandwidth is a valid multiplexing schedule. Since E-EDF already ensures that the receiver streaming buffer does not overflow, all we need to show is that no playback starvation occurs by following the greedy E-EDF multiplexing schedule.

Lemma 6 *A greedy E-EDF multiplexing schedule S_M^{geedf} under the inevitable bandwidth does not lead to playback starvation.*

Proof: Suppose S_M^{geedf} leads to playback starvation. Assume the first playback starvation occurred at time $t(j)$, when a media element from stream s_k with deadline $t(j)$ had not been delivered by time $t(j)$. Let $t(i)$ be the last time instant prior to $t(j)$ that either a media element with deadline later than $t(j)$ was being transmitted or no media element was under transmission. $t(i)$ is 0 is no such $t(i)$ exists. $t(i)$ must be a deadline of some media element since media elements with earlier deadlines only becomes eligible as the streaming buffer changes from full to non-full after a media element is picked up and rendered.

At time $t(i)$, we divide streams into two sets. The streams with media elements with deadlines $t(j)$ or earlier not yet delivered are classified as pending streams. The remaining are called non-pending streams. The only reason that either a media element with deadline later than $t(j)$ was being transmitted or no element was under transmission until $t(i)$ under E-EDF was that the pending streams (with elements of earlier deadlines) all had their streaming buffer filled until $t(i)$, thus making their media elements ineligible for transmission.

From time $t(i)$ to $t(j)$, only media elements belonging to pending streams got transmitted because $t(i)$ was the latest time a media elements with deadline later than $t(j)$ could be transmitted. The schedule S_M^{geedf} also consumed all the inevitable bandwidth from $t(i)$ to $t(j)$ because $t(i)$ was the last time that all media elements could be ineligible. The amount of data being transmitted from $t(i)$ to $t(j)$ is thus $t(j) - t(i)$ times the inevitable bandwidth in *Lemma 5*, resulting in an amount of $max_{0 \le i < j \le N} \sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}$.

At $t(i)$, all pending streams had their streaming buffer filled, $S_k^{geedf}(t(i)) = U_k(t(i))$ for all pending stream s_k, where S_k^{geedf} is the per-stream schedule for stream s_k. But by the earlier assumption, S_M^{geedf} caused a playback starvation at time $t(j)$. So the amount of data transmitted from $t(i)$ to $t(j)$ is less than $\sum_{s_k \in Pending} L_k(t(j)) - S_k^{geedf}(t(i))$, since only pending stream elements with deadlines no later than $t(j)$ were transmitted from $t(i)$ to $t(j)$. For every pending stream s_k, there existed media elements with deadlines $t(j)$ or earlier to be transmitted at time $t(i)$ and the streaming buffer was full at $t(i)$, so $L_k(t(j)) - U_k(t(i)) = L_k(t(j)) - S_k^{geedf}(t(i)) > 0$. For every non-pending stream s_k, media elements with deadlines $t(j)$ or earlier had already been transmitted at time $t(i)$, indicating $L_k(t(j)) - U_k(t(i)) \le L_k(t(j)) - S_k^{geedf}(t(i)) \le 0$. Rewriting the previous expression, the amount of data transmitted from $t(i)$ to $t(j)$ is less than $\sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}$.

This contradicts the earlier result of having to transmit $max_{0 \le i < j \le N} \sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}$ amount of media data from $t(i)$ to $t(j)$.

□

Combining the results obtained so far, we have the following main result.

Theorem 2 *The optimal multiplexing bandwidth (OMB) for a set of continuous media streams is as follows, $OMB = max_{0 \le i < j \le N} \sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}/(t(j) - t(i))$. Furthermore, the greedy E-EDF multiplexing schedules under the bandwidth OMB are a family of optimal multiplexing schedules achieving the optimal multiplexing bandwidth.*

Proof: By *Lemma 6* and *Theorem 1*. □

4.4 Multiplexing and Aggregation

In the last section, we introduced the concept of stream aggregation and showed that stream aggregation served as a bound for stream multiplexing. In particular, we showed that any valid stream multiplexing schedule is a valid stream aggregation schedule but the *vice versa* is not true in general. In the following, we show the exact, and somewhat interesting relationship between stream multiplexing and stream aggregation. Let Φ be the set of the continuous media streams, $\Phi = \{s_1, s_2, \ldots, s_N\}$.

Lemma 7 *Let S_M be a valid multiplexing schedule, it holds that $Peak(S_M) \ge Peak(S_\phi^*)$, for any $\phi \subseteq \Phi$, where S_ϕ^* is the optimal aggregation schedule for the set of streams ϕ.*

Proof: Recall that S_M is the sum of the per-stream transmission schedules $S_1, S_2, \ldots, S_K$ pertaining to the individual streams $s_1, s_2, \ldots, s_K$, respectively. So $S_M = \sum_{k=1}^{K} S_k \ge \sum_{s_k \in \phi} S_k$. Note that $\sum_{s_k \in \phi} S_k$ is a valid multiplexing schedule for the stream set ϕ, applying *Lemma 4* to the stream set ϕ, we have $Peak(\sum_{s_k \in \phi} S_k) \ge Peak(S_\phi^*)$. Combining the above, we get $Peak(S_M) \ge Peak(S_\phi^*)$. □

Lemma 8 *Let ϕ be a subset of streams, then for any $t(i) < t(j)$, $Peak(S_\phi^*) \ge \sum_{s_k \in \phi} (L_k(t(j)) - U(t(i)))/(t(j) - t(i))$.*

Proof: By *Lemma 2*, the feasible region for the stream aggregation of the stream set ϕ is $(\sum_{s_k \in \phi} L_k, \sum_{s_k \in \phi} U_k)$. Applying *Lemma 1* to the stream aggregation, $Peak(S_\phi^*) = \max\{\max_{t(i) < t(j)} (\sum_{s_k \in \phi} L_k(t(j)) - \sum_{s_k \in \phi} U_k(t(i)))/(t(j) - t(i)), 0\}$. Thus $Peak(S_\phi^*) \ge \sum_{s_k \in \phi} (L_k(t(j)) - U(t(i)))/(t(j) - t(i))$. □

Theorem 3 *The optimal multiplexing bandwidth (OMB) of a set of streams is the maximum of the optimal aggregation bandwidths among all its stream subsets,* $OMB(\Phi) = \max_{\phi \subseteq \Phi} Peak(S^*_\phi)$.

Proof: We first show $OMB(\Phi) \leq \max_{\phi \subseteq \Phi} Peak(S^*_\phi)$. Let $t(i)$ and $t(j)$ be such parameters that maximize the right side of the OMB formula in *Theorem 2*. In other words, $OMB(\Phi) = \sum_{k=1}^{K} \max\{L_k(t(j)) - U_k(t(i)), 0\}/(t(j) - t(i))$. Let ϕ be the set of such streams s_k that $L_k(t(j)) - U_k(t(i)) \geq 0$, the above equation becomes $OMB(\Phi) = \sum_{s_k \in \phi} L_k(t(j)) - U_k(t(i))/(t(j) - t(i))$. But the right side of the equation is exactly the one in *Lemma 8*, thus $Peak(S^*_\phi) \geq OMB(\Phi)$ using *Lemma 8*. Consequently, $OMB(\Phi) \leq Peak(S^*_\phi) \leq \max_{\phi \subseteq \Phi} Peak(S^*_\phi)$.

On the other hand, by *Lemma 7*, $OMB(\Phi) \geq Peak(S^*_\phi)$ for any stream subset $\phi \subseteq \Phi$. It follows that $OMB(\Phi) \geq \max_{\phi \subseteq \Phi} Peak(S^*_\phi)$. □

The result has some interesting implications. First, it establishes a clear relationship between stream multiplexing and stream aggregations. On one hand, for the same set of streams, stream multiplexing is at least as bandwidth intensive as stream aggregation. On the other hand, stream multiplexing for a set of streams is no more bandwidth intensive than the stream aggregations of some of its stream subsets. Therefore, stream aggregations bound the bandwidth requirement of stream multiplexing both ways from upper and from under. In a sense, the bandwidth requirements of stream aggregation and stream multiplexing are roughly on the same "order".

Second, the result also suggests that the stream aggregation of a subset of streams could be strictly more bandwidth intensive than the stream aggregation of the entire set of the streams. The phenomenon has indeed been observed in some cases during our experiments using real compressed video traces. In most of our experiments, the whole-set aggregation turned out to be the most dominant aggregation as would usually be expected. The apparent anomaly could possibly be attributed to the fact that a larger aggregation is also associated with a large streaming buffer, which could be used effectively to reduce its bandwidth requirement. In the cases that a subset aggregation dominates, multiplexing the whole set of streams requires the same amount of bandwidth as multiplexing a specific subset of the streams. In some sense, the rest of the streams are being delivered "for free", without incurring additional bandwidth.

5 Mitigating Adverse Per-Stream Effects

Although optimal multiplexing maximally reduces the overall bandwidth requirement for a set of continuous media stream on a shared network path (or disk, network interface etc.), it can have adverse effects on the individual per-stream schedules. In particular, it can cause the bandwidth requirement of individual streams to go up. In extreme cases, the resulting per-stream bandwidth can become so high that the adverse effects on individual stream offset any multiplexing gains achieved using optimal multiplexing.

5.1 Worst-Case Per-Stream Bandwidth

In scenarios such as media-on-demand where continuous media streams are delivered over a server-rooted distribution tree to a set of distributed receivers (e.g. Figure 1), the bandwidth available for an individual stream is limited by the residual network path (e.g. from a local media reflector to the receiver) or the last hop after the streams destined to different receivers diverge. If the per-stream bandwidth resulted from optimal multiplexing is higher than the bandwidth available on the residual network path or the last hop, the schedule for this individual stream can not be satisfied by the available bandwidth on the path leading to the receiver and becomes infeasible. The server would have to deliver the stream separately without multiplexing with other streams.

In the worst case, the per-stream bandwidth can be as high as OMB, the total multiplexing bandwidth. An example is shown in Figure 7. Greedy E-EDF multiplexing requires the full bandwidth OMB be consumed whenever there are eligible media elements to transmit. In addition, E-EDF mandates that media elements with later deadlines never receive transmission until eligible elements with earlier deadlines are fully transmitted. Consequently, if a media element from a continuous media stream is the only eligible media element left with the earliest deadline, it takes the entire bandwidth of OMB. The resulting schedule for that particular stream thus has a peak bandwidth requirement of OMB. Even when the continuous media streams are fully synchronized (e.g. have identical frame rate and start time) so that there exist multiple media elements with the same deadline with the possibility to shared the total bandwidth, the worst-case per-stream bandwidth of OMB can still occur as the schedule progresses.

Compounding the problem is that OMB increases with the number of continuous media streams. When there are a large number of streams to be multiplexed together, the worst-case per-stream bandwidth requirement of OMB could be too high to be satisfied by the bandwidth resource available to an individual stream.

5.2 Equivalent Multiplexing Schedules

To mitigate the high per-stream bandwidth effect described above, we introduce a technique that results in a family of equivalent multiplexing schedules. These equivalent multiplexing schedules achieves the same optimal multiplexing bandwidth. However, they result in much lower per-stream bandwidths. The technique is based on a simple observation: for any continuous media stream, media elements transmitted between a deadline $t(i)$ and the next deadline $t(i+1)$ in a valid schedule can be rearranged in any arbitrary manner without sacrificing validity of the schedule.

Lemma 9 *Let* S_k *be a per-stream schedule of stream* s_k *as part of a valid multiplexing schedule, and* $t(i)$ *and* $t(i+1)$ *be two consecutive deadlines of stream* s_k. *The multiplexing schedule resulted from rearranging the transmission of media elements of stream* s_k *transmitted between* $t(i)$ *and* $t(i+1)$ *in* S_k *remains a valid multiplexing schedule.*

Proof: It suffices to show the rearranged per-stream schedule S'_k is valid in the interval $[t(i), t(i+1)]$. As elaborated earlier, $S'_k(t) = S_k(t)$, for $t = t(i)$ and $t = t(i+1)$, since the rearrangement is restricted to the interval $[t(i), t(i+1)]$. In the interior of the interval, $\forall t \in (t(i), t(i+1))$, $L(t) = L(t(i))$ and $U_k(t) = U_k(t(i+1))$ by the fact that $t(i)$ and $t(i+1)$ are consecutive deadlines. Thus for all $t \in (t(i), t(i+1))$, $S'_k(t) \leq S'_k(t(i+1)) = S_k(t(i+1)) \leq U_k(t(i+1)) = U_k(t)$ and $S'_k(t) \geq S'_k(t(i)) = S_k(t(i)) \geq L_k(t(i)) = L_k(t)$, which shows S'_k is valid. □

Given a certain amount of media data to be transmitted between two consecutive deadline points for a given stream, the transmission can be *evenly* spread out throughout the entire interval. The deadlines can also belong to different

streams, as long as they are adjacent. An illustrative example is given in Figure 7.

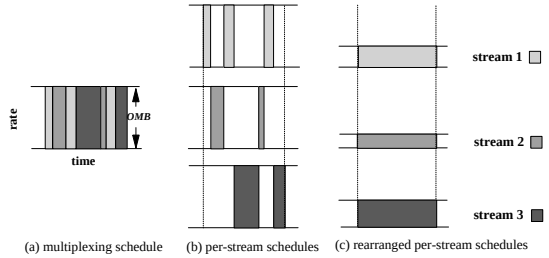


Figure 7: Inter-deadline transmission rearrangement

We examine the effect of rearrangement on the overall multiplexing schedule. Note that the transmission of all media data in a time interval are evenly spread out in the given time interval. But the total amount of media data delivered over the interval does not change, so the rearranged multiplexing schedule with even-rate transmission have a bandwidth no more than the original multiplexing schedule in this interval. This is true for every consecutive deadline interval, thus we have the following lemma.

Lemma 10 *The even inter-deadline spread transmission rearrangement does not result in a bandwidth increase of the multiplexing schedule.*

Proof: By the preceding argument. □

We apply the technique to the greedy E-EDF multiplexing schedules under bandwidth OMB. Because the greedy E-EDF schedules are optimal, the resulting schedules must also be optimal by *Lemma 10*. Consequently, the technique reduces the per-stream bandwidth requirements while maintaining optimal multiplexing.

Theorem 4 *The multiplexing schedules resulted from applying even inter-deadline spread technique to the greedy E-EDF schedules under bandwidth OMB are optimal and achieve the optimal multiplexing bandwidth OMB.*

Proof: First the resulting multiplexing schedules are valid based on *Lemma 9*. By *Lemma 10*, the resulting schedules requires no more bandwidth than greedy E-EDF schedules. But by *Theorem 2*, greedy E-EDF schedules are optimal and achieves the lowest possible bandwidth OMB. □

5.3 Algorithm and Complexity

The following algorithm implements the even inter-deadline spread technique. Given an arbitrary bandwidth bound B (in the case of optimal multiplexing, $B = OMB$), the algorithm calculates the per-stream schedules by applying the even inter-deadline spread technique to the greedy E-EDF schedule under bandwidth B.

Algorithm to calculate per-stream schedules under bandwidth B

Function $Per\_Stream\_Schedules(B)$
  d_1 = first deadline;
  $Residual\_Buffer[k] = Stream\_Buffer\_Size[k]$, $\forall k = 1, 2, \ldots, K$;
  $Eligible\_Set$ = {all media elements};
  While (d_1 is not the final deadline)
    d_2 = next deadline after d_1;
    $Total\_Data\_To\_Transmit = B * (d_2 - d_1)$;
    $Eligible\_Set = Eligible\_Set \bigcup$ {new eligible elements at d_1};
    $Stream\_Data\_To\_Transmit[k] = 0$, $\forall k = 1, 2, \ldots, K$;
    While ($Data\_To\_Transmit > 0$ and $Eligible\_Set \neq \emptyset$)
      o_i = element with the earliest deadline in $Eligible\_Set$;
      s_k = the stream to which o_i belongs;
      $Element\_Data\_To\_Transmit =$
        $\min\{s(i), Residual\_Buffer[k], Total\_Data\_To\_Transmit\}$;
      $s(i)$ −= $Element\_Data\_To\_Transmit$;
      If $s(i) = 0$ $Eligible\_Set = Eligible\_Set$ - $\{o_i\}$;
      $Stream\_Data\_To\_Transmit[k]$ += $Element\_Data\_To\_Transmit$;
      $Total\_Data\_To\_Transmit$ −= $Element\_Data\_To\_Transmit$;
      $Residual\_Buffer[k]$ −= $Element\_Data\_To\_Transmit$;
      If $Residual\_Buffer[k] = 0$ $Eligible\_Set =$
        $Eligible\_Set$ - {elements of stream s_k};
    End
    Transmit stream s_k at rate $Stream\_Data\_To\_Transmit[k]/(d_2 - d_1)$,
      $\forall k = 1, 2, \ldots, K$;
    $d_1 = d_2$;
  End
End

The algorithm performs one sweep of the media elements in the order of their deadlines. Between two consecutive deadlines, it executes the greedy E-EDF multiplexing scheme under the bandwidth B, and calculates the amount of media data from each of the streams to be transmitted. The total quota of media data to be transmitted during a deadline interval $[d_1, d_2]$ is $B * (d_2 - d_1)$ by the greedy criterium, provided there are always media elements eligible. The algorithm proceeds to distribute this quota to the each of the media streams. The process is carried out on the set of eligible media elements in the increasing order of their deadlines. The earliest eligible media element is allocated a portion of the transmission quota equal to its size. The quota distribution continues until it is fully distributed or no media element is eligible. Subsequently, the aggregated per-stream transmission quota is spread out evenly throughout the deadline interval $[d_1, d_2]$. With proper data structure, the algorithm takes linear time $O(KN)$ (in the number of media elements).

5.4 Optimal Admission Control

In addition to mitigating the per-stream effect of optimal multiplexing, the above per-stream scheduling algorithm plays another important role. The algorithm can be used as *admission control* for on-demand continuous multimedia streaming. Let the media server have a bottleneck bandwidth of B and is serving a set of on-going media streams. When a new media streaming request arrives, the server has to decide whether it has enough capacity (bandwidth) to accommodate the new stream. To make the admission decision, we run the algorithm on the new set of streams, with the new stream added, under the total bandwidth B. The new stream request is admitted if no playback starvation occurs (easily checked against the lower bounds L_k) during the execution of the algorithm. Consequently, the algorithm accomplishes the dual tasks that are essential for the media server: admission control and per-stream transmission scheduling.

The proposed admission control algorithm is in fact an optimal admission controller, based on the optimality of the inter-deadline spread technique. It admits the maximum number of media streams that can be supported, covering the largest admissible region.

6 Numeric Results

We measure the performances of three different stream multiplexing schemes using real compressed video traces. The

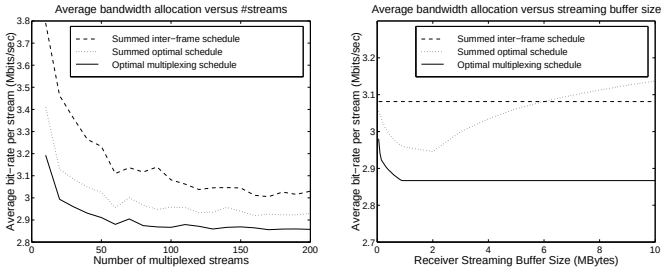
Video Name	Avg. Rate (Mbps)	Frame Sizes (Byte)			
		Avg	Max	Min	S. Dev.
Beauty and Beast	3.04	12661	30367	2701	3580
Big	2.96	12346	23485	1503	2366
Crocodile Dundee	2.59	10773	19439	1263	2336
E.T.	2.17	9022	19961	2333	2574
Home Alone 2	2.73	11383	22009	3583	2480
Honey, I Blew Up the Kid	3.32	13836	23291	3789	3183
Hot Shots 2	3.06	12766	29933	3379	3240
Jurassic Park	2.73	11363	23883	1267	3252
Junior	3.36	14013	25119	1197	3188
Rookie of the Year	2.98	12435	27877	3531	2731
Sister Act	2.86	11902	24907	1457	2608
Sleepless in Seattle	2.28	9477	16617	3207	2459
Speed	2.97	12374	29485	2741	2707
Total Recall	2.88	11978	24769	2741	2692

Table 1: Motion-JPEG compressed video traces used in the experiments

three multiplexing schemes in our experiments are the *summed inter-frame schedule* and the *summed optimal schedule* defined in Section 3 and the *optimal multiplexing schedule* introduced in Section 4. Recall that the first two schemes are obtained by simply stacking the individual stream schedules together, one based on the simplistic inter-frame individual schedules and the other based on the more sophisticated optimal individual schedules. On the other hand, optimal multiplexing uses greedy E-EDF scheduling, taking advantage of both intra-stream and inter-stream correlations.

The variable-bit-rate (VBR) video traces used in the experiments were obtained from [3] and available on-line at [4]. The traces are compressed using Motion-JPEG video coding standard, with the statistics of the video traces shown in Table 1. (We also used MPEG video traces and mixed M-JPEG and MPEG traces, the results are similar and not shown here due to space reasons).

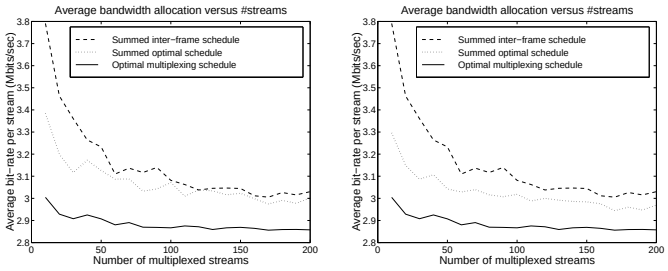
To produce a sufficient number of concurrent video streams, we use a procedure to extract 30-minute long video segments from the available traces. For example, in order to generate a certain number of video streams to multiplex, we cycle through the available video traces and extract 30-minute video streams at equally-distanced points within each of the video traces. The generation process actually reflects a somewhat realistic scenarios where users started viewing at different times, thus are viewing different parts of the same programs at the same time.



(a) 10-200 streams, 1MB buffer (b) 100 streams, 50KB-10MB buffer

Figure 8: Multiplexing performances with homogeneous streaming buffers

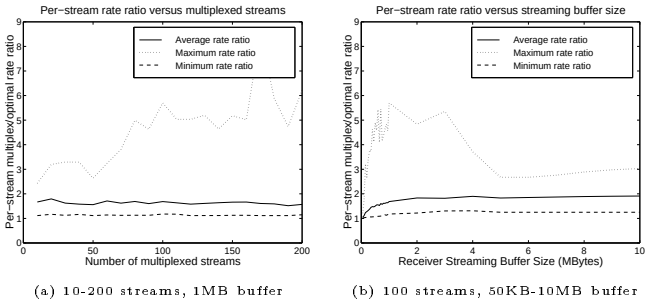
We measure the performances of the multiplexing schemes based on the average bandwidth allocation per multiplexed stream (i.e. the total bandwidth divided by the number of multiplexed streams). The schedules are calculated based on a one-second startup delay to avoid the possible startup delay effect on the overall bandwidth [18]. Figure 8(a) shows the multiplexing performances of 10 to 200 video streams, generated using the above procedure, with a 1MB streaming buffer at the receiver. As expected, optimal multiplexing achieves the lowest bandwidth among the three multiplexing schemes with any number of multiplexed streams (in fact, it achieves the bandwidth lower-bound). Figure 8(b) shows the multiplexing of 100 streams with varying streaming buffer sizes from 50KB to 10MB. Somewhat surprising in this figure is the fact that the rudimentary summed intra-frame schedule sometimes out-performs the more sophisticated summed optimal schedule, which demonstrates the fact that optimally scheduling individual streams may not always result in a better overall schedule.



(a) 100KB and 10MB buffer (b) 100KB, 1MB and 10MB buffer

Figure 9: Multiplexing performances with heterogeneous streaming buffers

In addition to homogeneous streaming buffer sizes, we also tested the multiplexing performances under heterogeneous receiver streaming buffer sizes. Figure 9(a) shows the multiplexing of 10 to 200 streams with two distinct levels of streaming buffer sizes: 100KB and 10MB. Half the receivers have 100KB streaming buffers and the other half have 10MB ones. Figure 9(b) shows the performances with three distinct streaming buffer levels: 100KB, 1MB and 10MB. The results are similar to the ones with homogeneous buffer sizes, thus the multiplexing techniques apply to heterogeneous environments as well.



(a) 10-200 streams, 1MB buffer (b) 100 streams, 50KB-10MB buffer

Figure 10: Per-stream bandwidth effect using transmission rearrangement

Finally, we tested the performances of mitigating the adverse per-stream bandwidth effect using inter-deadline transmission rearrangement. We used the same parameters as before, 10 to 200 streams with 1MB streaming buffers and 100 streams with 50K to 10MB streaming buffers. Applying the inter-deadline transmission rearrangement tech-

nique, the per-stream schedules were derived and recorded. The peak rates of each per-stream schedule are then represented as ratio versus the optimal peak rates of the respective stream scheduled individually. For any given number of multiplexed streams, we show the average, maximum and minimum of such ratios in Figure 10. The results show that the technique performs reasonably well. Comparing to the per-stream bandwidths as high as the optimal multiplexing bandwidth (OMB) of the whole set of streams prior to rearrangement, the reduction is significant. There could still be room for improvement, which is part of the future research.

7 Conclusion

We proposed bandwidth efficient multiplexing techniques for the delivery of continuous media streams to a set of distributed and heterogeneous receivers. In particular, we introduced a multiplexing scheme that is provably optimal, achieving the highest possible multiplexing gain, while supporting stringent and deterministic quality of service guarantees. The optimal multiplexing bandwidth was derived and expressed in closed-form. An algorithm for its calculation runs in linear time in most cases, which was not described here in detail due to space reasons. Given the optimal multiplexing bandwidth, a transmission rearrangement technique was introduced that resulted in reducing the per-stream bandwidth significantly, while maintaining multiplexing optimality. The reduction in per-stream bandwidth is necessary to mitigate the adverse effects of optimal multiplexing on the end-to-end per-stream bandwidth requirement. The linear-time per-stream scheduling algorithm was easily extended for admission control at the media server for on-demand continuous media streaming. The admission control is optimal in the sense that it supports the maximum number of continuous media streams.

Future research directions include studying a more general and potentially more difficult problem: what is the optimal multiplexing bandwidth achievable under specific per-stream bandwidth constraints. The result will be very useful when the available per-stream bandwidth resource is known in advance. Other future directions include improving the running time of on-demand admission control and per-stream scheduling, possibly through incremental techniques, and trying to design an theoretically-provable linear-time algorithm to calculate the optimal multiplexing bandwidth under general condition.

References

[1] K. Almeroth and M. Ammar. The Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service. *IEEE Journal on Selected Areas in Communications*, 14(6):1110–1122, Aug. 1996.

[2] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *Proceedings of ACM Multimedia'94*, pages 391–398, Oct. 1994.

[3] W. Feng. Video-on-Demand Services: Efficient Transportation and Decompression of Variable Bit Rate Video. *Ph.D. Dissertation*, Apr. 1996.

[4] W. Feng. The Ohio State/Univ. of Michigan Video Movie Library. http://www.cis.ohio-state.edu/~wuchi/Video.

[5] W. Feng and S. Sechrest. Smoothing and Buffering for the Delivery of Prerecorded Compressed Video. *Proceedings of the IS&T/SPIE Multimedia Computing and Networking*, pages 234–244, Feb. 1995.

[6] M. Garrett and W. Willinger. Analysis, Modeling and Generation of Self-Similar Video Traffic. In *Proceedings of the ACM SIGCOMM'94*, Sept. 1994.

[7] L. Golubchik, J. Lui, and R. Muntz. Adaptive Piggybacking: A Novel Technique for Data Sharing in Video-On-Demand Storage Servers. *ACM Multimedia Systems Journal*, 4(3):140–155, 1996.

[8] M. Krunz, W. Zhao, and I. Matta. Scheduling and Bandwidth Allocation for Distribution of Archived Video in VoD Systems. *Journal of Telecommunication Systems, Special Issue on Multimedia*, 9(3,4), Sept. 1998.

[9] D. Lee and F. Preparata. Euclidean Shortest Path in the Presence of Rectilinear Barriers. *Networks*, 14:393–410, 1984.

[10] D. LeGall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, Apr. 1991.

[11] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment. *Journal of the ACM*, 20(1):46–61, Jan. 1973.

[12] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proceedings of the ACM SIGCOMM'96*, Aug. 1996.

[13] M. Naghshineh and M. Willebeek-LeMair. End-to-End QoS Provisioning in Multimedia Wireless/Mobile Networks Using an Adaptive Framework. *IEEE Communications Magazine*, pages 72–81, Nov. 1997.

[14] J. Salehi, Z. Zhang, J. Kurose, and D. Towsley. Supporting Stored Video: Reduce Rate Variability and End-to-End Resource Requirements through Optimal Smoothing. *IEEE/ACM Transactions on Networking*, 6(4), Aug. 1998.

[15] J. Salehi, Z. Zhang, J. F. Kurose, and D. Towsley. Supporting Stored Video: Reduce Rate Variability and End-to-End Resource Requirements through Optimal Smoothing. In *Proceedings of ACM SIGMETRICS'96*, 1996.

[16] Z. Zhang, J. Kurose, J. Salehi, and D. Towsley. Smoothing, Statistical Multiplexing and Call Admission Control for Stored Video. *IEEE Journal on Selected Areas in Communications*, Aug. 1997.

[17] W. Zhao, M. Krunz, and S. K. Tripathi. Efficient Transport of Stored Video Using Stream Scheduling and Window-Based Traffic Envelopes. In *Proceedings of Internaional Conference on Communications (ICC'97)*, 1997.

[18] W. Zhao, T. Seth, M. Kim, and M. Willebeek-LeMair. Optimal Bandwidth/Delay Tradeoff for Feasible-Region-based Scalable Multimedia Scheduling. In *Proceedings of the IEEE INFOCOM'98*, 1998.