# A Two-Tiered On-Line Server-Side Bandwidth Reservation Framework for the Real-Time Delivery of Multiple Video Streams

Jorge M. Londoño and Azer Bestavros

Computer Science Department, Boston University, Boston, MA, USA

July 1st, 2008

BUCS-TR-2008-012

## ABSTRACT

The advent of virtualization and cloud computing technologies necessitates the development of effective mechanisms for the estimation and reservation of resources needed by content providers to deliver large numbers of video-on-demand (VOD) streams through the cloud. Unfortunately, capacity planning for the QoS-constrained delivery of a large number of VOD streams is inherently difficult as VBR encoding schemes exhibit significant bandwidth variability. In this paper, we present a novel resource management scheme to make such allocation decisions using a mixture of per-stream reservations and an aggregate reservation, shared across all streams to accommodate peak demands. The shared reservation provides capacity *slack* that enables statistical multiplexing of peak rates, while assuring analytically bounded frame-drop probabilities, which can be adjusted by trading off buffer space (and consequently delay) and bandwidth. Our two-tiered bandwidth allocation scheme enables the delivery of any set of streams with less bandwidth (or equivalently with higher link utilization) than state-of-the-art deterministic smoothing approaches. The algorithm underlying our proposed framework uses three per-stream parameters and is linear in the number of servers, making it particularly well suited for use in an on-line setting. We present results from extensive trace-driven simulations, which confirm the efficiency of our scheme especially for small buffer sizes and delay bounds, and which underscore the significant realizable bandwidth savings, typically yielding losses that are an order of magnitude or more below our analytically derived bounds.

**Keywords:** Video on demand, quality of service, admission control, streaming media

## 1. INTRODUCTION

**Motivation:** On-demand video streaming over the Internet is here to stay. Over the years much research has been devoted to resource management issues of Internet video streaming from the perspectives of the client and the network. From the client's perspective, packets traversing the network may suffer from losses and high delay variability. Delay variability is typically managed by buffering a short period of the stream on the client side[1] to avoid *underflow* conditions which would lead to interruptions of the playback. Proper sizing of the client-side buffer is important because if underestimated, it would lead to *overflow* conditions whereby packets are lost with great detriment to playback quality. From the network's perspective, the high variability of the video streaming rates makes it very difficult to provision adequate bandwidth, while achieving either high utilization or low delay and buffer requirements. To that end, smoothing techniques[2] provide the means to achieve higher utilization by trading off a small amount of buffering/delay.

While significant attention has been devoted to the allocation of client and network resources in support of Internet video streaming applications, less attention has been given to the allocation of server-side resources. This is primarily due to the fact that, traditionally, such resources were assumed to be dedicated, well-provisioned (if not over-provisioned) server farms, for example. However, the advent of virtualization and cloud computing technologies requires us to revisit such assumptions, and consequently to develop effective mechanisms for the estimation and reservation of the *server-side* resources needed by content providers to deliver large numbers of video-on-demand (VOD) streams through the cloud. In this paper, we focus on the (cloud/virtual) server-side upload bandwidth needs necessary to service a large number of

Jorge Londoño / E-mail: jmlon@cs.bu.edu
Azer Bestavros / E-mail: best@cs.bu.edu

VOD streams subject to minimal QoS constraints. Moreover, since cloud resources are likely to be acquired (and paid for) dynamically in response to variable demand patterns, we focus our attention to approaches that are amenable to on-line/real-time use.

Clearly, the bandwidth allocation approaches used to provision network paths[1,3–6] can be used to provision the server bandwidth needs as well, but this would be inefficient resource-wise. In particular, the variability of VBR traffic patterns creates the opportunity for achieving better utilizations while guaranteeing the bandwidth demanded by individual streams, as proposed by Krunz et al.[7] Other proposals[8,9] advocate the use of adaptive encoding schemes so that the system may trade-off quality for bandwidth and CPU cycles.

In this paper we present an alternative scheme: by exploiting the opportunities for statistically multiplexing VBR streams at the server, we are able to achieve higher utilization while providing probabilistic bounds on dropped frames and deterministic bounds on delay and buffer requirements.

**Scope and Contributions:** We consider a VOD distribution system whereby a pool of acquired (cloud) servers must handle requests coming from a large population of clients. The system we envision has the following attributes:

1. On-line: The system has no *a priory* knowledge of service requests.
2. Real-Time: The system must make admission/allocation decisions within a bounded amount of time.
3. Quality-of-Service: The system must satisfy per-stream bandwidth, delay and loss requirements, which are necessary to meet user's perceived quality expectations.
4. Efficient and Scalable: The system must be able to handle a large number of streams and must ensure high bandwidth utilization.

To that end, we present a server-side bandwidth and buffer management scheme which gives deterministic bounds on delays, and provisions bandwidth at rates that are typically much lower than other well known techniques. In addition, we present an extention to our scheme which trades off a small loss rate for a highly effective statistical multiplexing of peak bandwidth requirements of streams that are co-located on the server. Our scheme can be seen as providing a framework that enables a spectrum of design choices ranging from purely best-effort to exclusive reservations, such that the loss probabilities are bounded according to the system parameters. Within this framework, we present an on-line admission/allocation algorithm which uses three pre-computed parameters per stream to characterize the stream's bandwidth needs. Upon arrival of a request for video stream delivery, our admission/allocation algorithm makes its decision in time linear in the number of servers.

We present an extensive evaluation of our proposed approach using trace-driven simulations over a large collection of H.264 and MPEG4 video streams. The results we obtain confirm the efficiency and scalability of our scheme and underscore the significant realizable savings, typically yielding losses that are an order of magnitude or more below our analytically derived bounds.

## 2. A MULTI-STREAM BANDWIDTH ALLOCATION FRAMEWORK

### 2.1 Background and Basic Approaches

Video encoding standards, such as MPEG-2/4 or H.264, make use of three types of frame encodings: I-frames (interframe) contain a complete encoded frame, P-frames (predictive) which depend on the information of a previous I or P frame to be decoded, and B-frames (bidirectional) which require information of preceding and future frames in order to be decoded. The exact sequence of I, P and B frames used by a stream is called the pattern, and their transmission order is not necessarily the playback order. Typically, the pattern contains a single I frame and many P and B frames. The set of frames corresponding to one iteration of the pattern is called the Group of Pictures (GoP).* Figure 1 illustrates the cumulative number of bits, $A(t)$ for a sample GoP from a real VBR stream. The pattern is also indicated on the top.

*For a review of video encoding techniques see[10]

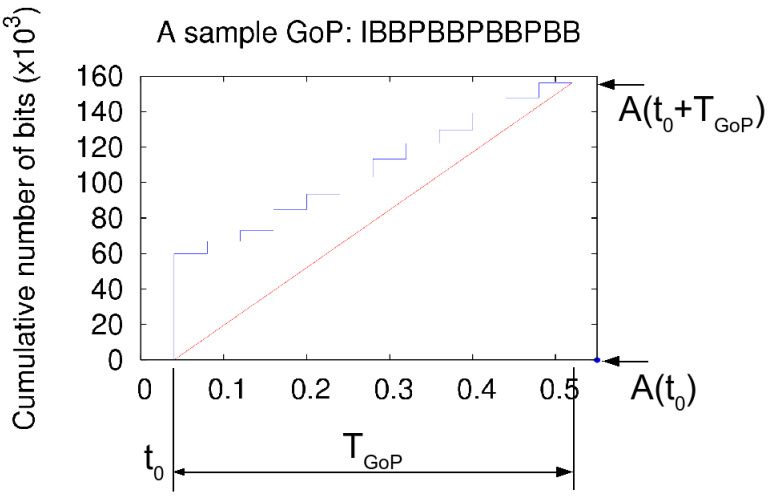



Figure 1. Temporal representation of a Group of Pictures (GoP)

In current standards the frame rate is constant, and two important parameters describing the stream are the mean bitrate and the peak bitrate defined as follows:

$$\text{Mean Rate} = \frac{a(t_f)}{t_f} \tag{1}$$

$$\text{Peak Rate} = \max_i \left\{ \frac{a(t_{i+1}) - a(t_i)}{t_{i+1} - t_i} \right\} \tag{2}$$

where $t_i$ indicates the time of the $i^{th}$ frame and $t_f$ is the time of the last frame. (It is assumed the stream starts a $t_1 = 0$).

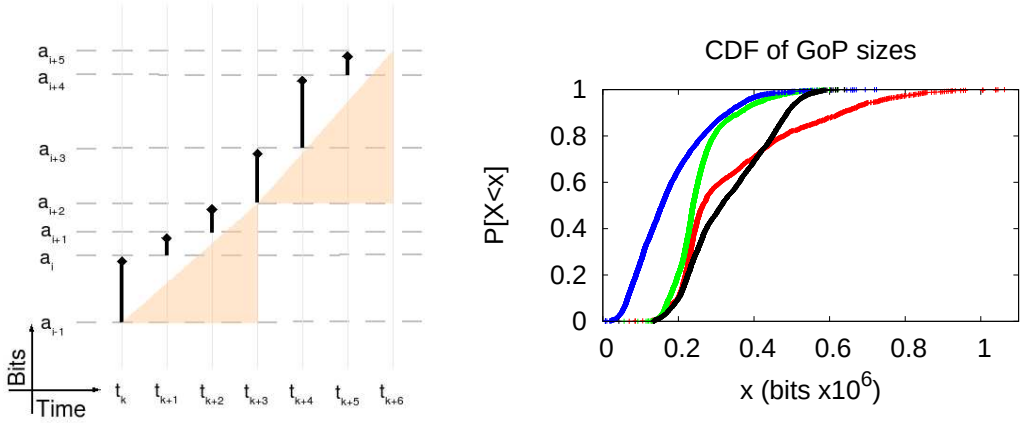
Using the stream's mean rate for reserving resources is not practical as it does not provide a small enough bound on the playback delay and, potentially, may require a very large buffer to avoid buffer underruns at the receiver. On the other hand, while using the peak rate would give the minimum playback delay and would minimize the amount of buffering required, it is also wasteful of resources as bandwidth utilization will be very low, making it impossible to scale the system to a large number of streams.

Previous work[3] uses the concept of *effective bandwidth* as a way to characterize (using a tight constant rate envelope) any time interval of the stream, so that buffering delay experienced by any packet during this interval is bounded. Together, the rate and the time bound determine the maximum buffer space needed. Although this effective bandwidth envelope method provides a deterministic, tight characterization of the stream, its direct application to reserve resources at the server imposes an exclusive reservation mechanism that undermines the possibility of improved scaling that could be achieved by statistically multiplexing many streams. A secondary drawback of this technique is that it is computationally expensive as it is *quadratic* in the number of frames in the stream.

## 2.2 Efficient Characterization of a Single VBR Video Stream

We propose a variation of the effective bandwidth model that allows computing the bandwidth and required buffer space in a single pass, *i.e.*, *linear* in the number of frames in the stream, while still providing guaranteed bounds on the delay experienced by packets in the stream. We assume that the time is discretized at time instants $t_k = k \cdot T_{GoP}$ and we discretize the cumulative bits function as $a_k = a(t_{k+1})$, *i.e.* presenting all the GoP's bits at the beginning of the interval. Figure 2a illustrates this discretized version of the stream.

Let the *smoothing period* $T$ indicate the length of the time intervals we will smooth. In practice, it is convenient to make $T = sT_{GoP}$, *i.e.* an integral multiple of the GoP period, but this not required by the analysis that follows. By grouping the bits in the interval $T$, it is possible to smooth the peaks present in the interval. We define the smoothed-bandwidth $\varepsilon$ as the




(a) (b)

Figure 2.

maximum bandwidth required over all discrete groups of length $T$:

$$\varepsilon = \max_k \left\{ \frac{A(t_k + T) - A(t_k)}{T} \right\}. \tag{3}$$

Figure 2a illustrates two such groups for $T = 3T_{GoP}$ (the shadowed triangles). An important consequence of this definition is that at times $t_k = nT$ (multiples of $T$), it is guaranteed that the transmission buffer will be empty, and thefore this gives an upper bound of $T$ on the delay experienced by any packet belonging to this interval. For the same reason, the maximum buffer space required is

$$\beta = \varepsilon T. \tag{4}$$

Clearly, equation (3) can be computed in a single pass over the stream and then equation (4) is a single operation. Therefore, the computation is *linear* on the number of frames.

### 2.3 A Two-Tiered Bandwidth Allocation Framework

Our second step is to relax the deterministic guarantee in equation (3). It is well known[10–12] that the distribution of frame sizes exhibits long range dependence. In other words, there is a non-negligible probability of having very large frames, although the distribution itself is skewed towards small frame sizes. Consequently, smoothing out a large frame may require relatively long period. Even worse, large frames do not occur independently, but they tend to appear clustered in time as evidenced by the autocorrelation of GoP sizes.[10] Figure 2b illustrates the CDF of some actual video traces providing evidence of the large mass of small frame sizes as well as the long tails.

Our proposed approach is not to use the maximum bandwidth for reservation, as this value is very unlikely to occur. Instead, we propose the two-tiered bandwidth reservation approach detailed below:

(a) Each stream has a guaranteed reservation for an amount equal to $\varepsilon^{(p)}$, which denotes the $p$ percentile of the distribution of the values $(A(t_k + T) - A(t_k))/T$.

(b) Each set of co-located streams is assigned an additional *slack* reservation for an amount equal to $\eta$. This reserved bandwidth is *shared* by all the streams. When an instantaneous burst requirement of a stream exceeds its reservation, that stream may use the shared slack in $\eta$ to accommodate that burst. This enables the statistical multiplexing of the peaks of various streams within the shared slack, effectively providing a cushion to insure against individual streams exceeding their reservations.

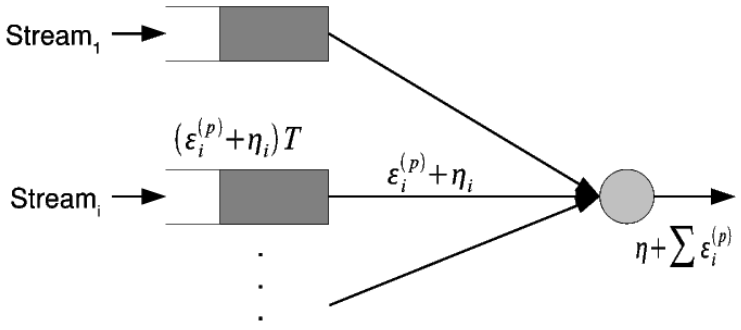



Figure 3. Architecture of video streaming system

## 2.4 Architecture of the Video Streaming System

Figure 3 illustrates the architecture of a video streaming system based on the two-tiered bandwidth allocation framework presented above. The system reserves a fixed amount $\varepsilon_i^{(p)}$ of bandwidth per stream, and also and additional shared bandwidth of $\eta$, which can be arbitrarily distributed during each smoothing period $T$, so that each stream receives additional capacity $\eta_i$, but preserving the total $\sum \eta_i \leq \eta$. The buffer capacity is also dynamically adjusted for each smoothing period. Basically, a stream may use up to $(\varepsilon_i^{(p)} + \eta_i)T$ buffer bits. As a consequence of these settings, all the bits in the buffer will be delivered by the end of the smoothing period if the output line is never idle. In other words, the maximum bit latency is $T$.

Special care must be given to ensure that the output line is never idle. Our subsequent analysis depends on having an empty buffer at the end of each smoothing cycle. Of course, this condition can be easily enforced in practice. In the case of stored media, all the system has to do is to prefetch all the frames belonging to the smoothing period. In the case of a live-feed, this condition may be ensured by delaying the stream by $T$.

The exclusively reserved portion of the bandwidth guarantees the timely delivery of a fraction $p$ of the GoPs. It is also possible to give a probabilistic guarantee for the remaining $1 - p$ fraction as follows: Let $X_i$ be the random variable representing the total number of bits in an interval of size $T$ of the $i^{th}$ stream at some point in time $t_k$. If $X_i < \varepsilon^{(p)} \cdot T$, then the reserved bandwidth is enough, otherwise the excess amount will have to be handled using the shared slack. Let us call $Y_i$ the excess amount per stream, so that

$$Y_i = \begin{cases} X_i - \varepsilon^{(p)} \cdot T & \text{if } X_i > \varepsilon^{(p)} \cdot T \\ 0 & \text{otherwise} \end{cases} .$$

The total slack $\eta$ will be sufficient if $\sum Y_i \leq \eta \cdot T$. Otherwise, we simply drop as many frames as necessary during this interval. Without any additional knowledge of the distribution of sizes and without any assumptions on the independence of the streams, the Markov's inequality allows us to bound the loss probability as follows.

$$\begin{aligned} P\left[\sum Y_i > \eta \cdot T\right] &\leq \frac{E\left[\sum Y_i\right]}{\eta \cdot T} \\ &\leq \frac{\sum E\left[Y_i\right]}{\eta \cdot T}, \end{aligned} \tag{5}$$

where the values $E\left[Y_i\right]$, which depend only on the value of $p$, can be easily precomputed for each stream. Although this expression could be used to estimate the slack $\eta$ given a bound on the loss probability, the value thus obtained is too large for practical purposes. Instead, we will use this equation to give a guaranteed bound, but leave open the actual computation of $\eta$ to heuristics that may perform well in practice. In particular for the video streaming application we will present in the next section the *maximum rule* gives much lower looses in practice (as we will show in the experimental section).

It is important to note that our framework enables a wide spectrum of possibilities. On the one hand, by setting $p = 0$, there would be no per-stream reservations ($\varepsilon^{(p)} = 0$), corresponding to a best-effort system with capacity $\eta$. On the other

hand, by setting $p = 1$, the per-stream reservation would be $\varepsilon^{(p)} = \varepsilon$, yielding $E[Y_i] = 0$, meaning that there are no losses at the source, and of course no need for slack reservation.

Alternatively, our framework can be seen as enabling an optimization problem, whereby the valuable resource is the bandwidth, and the total bandwidth $\eta + \sum_i \varepsilon_i^{(p)}$ may be minimized through an appropriate choice of $p$, subject to constraints (*e.g.*, on the maximum loss probability or other metrics).[†]

### 2.5 Admissibility Test

We conclude this section by presenting a strategy for admitting a new video stream $i$ to one of the set of servers in the system. We assume that the system consists of a farm (cloud) of $n$ servers with bandwidth capacity $C_j$ for server $j$. We assume that the target upper bound on the loss probability per stream is $l$. Also, we assume that the values $\varepsilon$, $\varepsilon^{(p)}$, and $E[Y_i]$ have been precomputed in advance for each one of the streams in the system.

1. For the new the stream $i$, compute the slack
$$\eta_i = \varepsilon - \varepsilon^{(p)}.$$
2. Using the maximum rule, the new slack of server $j$ ($\eta'_j$) is the maximum of its current slack ($\eta_j$) and the slack of stream $i$.
$$\eta'_j = \max\{\eta_i, \eta_j\}.$$
3. Stream $i$ is admissible to (*i.e.*, could be served by) server $j$ if
$$\varepsilon_i^{(p)} + \eta'_j + \sum_{k \in j} \varepsilon_k^{(p)} \leq C_j$$
and
$$\frac{E[Y_i] + \sum_{k \in j} E[Y_k]}{\eta'_j \cdot T} \leq l.$$
4. If stream $i$ is not admissible to any one of the $n$ servers already in the system, then the system must acquire (from the cloud) a new server $n + 1$ or else the stream $i$ is rejected.

Clearly, the above is an on-line strategy. In particular, it is only necessary to keep three state variables per server: The current maximum $\eta_j$, the sum $\sum \varepsilon_k^{(p)}$ and the sum $\sum E[Y_k]$. Upon arrival of a stream the new maximum and the sums can be easily updated to check feasibility. For this reason, the total cost of admitting a new stream is linear in the number of servers.[‡]

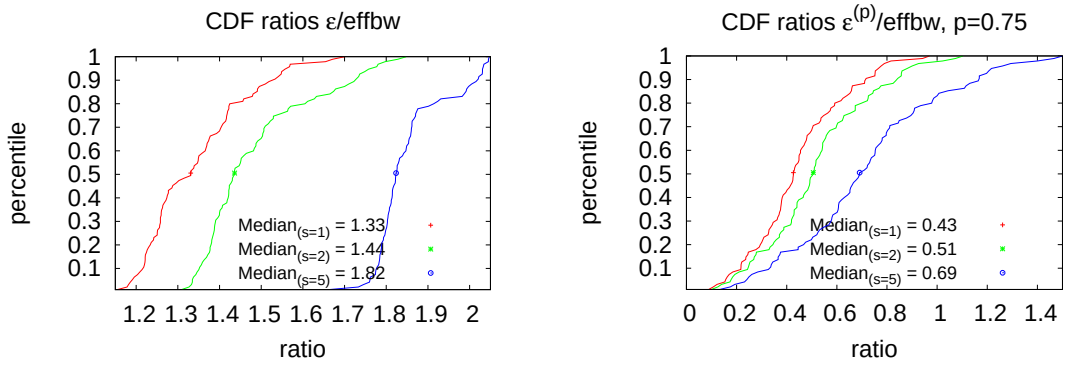
Notice that using the above admission test, there may be more than one server that is able to "host" the incoming stream $i$. This introduces a "server selection" problem. In particular if the farm is composed of heterogeneous servers, each of which with a different associated cost, then minimizing the total cost accrued by the system is akin to an extension of the *weighted-bin-packing* problem, and approximation techniques such as *first-fit* or *best-fit* could be used to select the server.

## 3. EXPERIMENTAL EVALUATION

In our experimental evaluation we used a collection of traces from.[10,13–17] These traces provide information about the frames of a large collection of video streams encoded with H.264 and MPEG-4 encoders, under a wide range of encoder configurations. We conducted our experiments with a subset of 95 streams, all of them with durations of half an hour or longer.

[†]The impact of the parameter $p$ is considered later in Section 3.3.

[‡]If we assume that streams leave (terminate), then keeping track of the maximum per server would require keeping the list of individual $\eta_j$ and finding the maximum after each departure/arrival would be $O(\log m)$ (using a heap), being $m$ the number of streams in a server.




(a) $\varepsilon$ to effective-bw ratio

(b) $\varepsilon^{(p)}$ to effective-bw ratio

Figure 4. CDF of the ratios of smoothed to effective bw

### 3.1 Comparison with Effective Bandwidth Allocation Strategy

To give a fair comparison with respect to the *effective bandwidth* metric, we set the delay bound to be the same in both cases and equal to an integral number $s$ of GoP periods. We then compute the effective bandwidth using the algorithm from,[3] and then compute the smoothed bandwidth $\varepsilon$ using eq. 3. By computing the ratio of the smoothed bandwidth to the effective bandwidth for each stream, we obtain the results shown in Figure 4a.

In almost all cases this ratio is well below two, but greater than one. This is expected as the smoothing technique does not provide an envelope as tight as the effective bandwidth approach. As the smoothing period ($sT_{GoP}$) increases, the ratio is larger, but the distribution becomes more steep as well, and the transition between the main body and the tail of the distribution becomes sharper at about $80\%$, indicating that for the majority of streams the ratio is below $1.85$, and for the remaining fraction it can go as high as $2.05$.

Similarly, Figure 4b shows the ratios of $\varepsilon^{(p)}$ to the effective bandwidth, computed for the case of $p = 0.75$. For the large majority of streams this ratio is well below one, indicating the possibility of bandwidth savings when using the $p$-percentile of the distribution (and of course, the $1 - p$ fraction will be handled using the shared slack later). Observe also that the ratio is smaller for small values of $s$, *i.e.* when the bound on the delay is smaller as in these cases the effective bandwidth technique needs more bandwidth to maintain the bound on the delay. This highlights the potential of our technique for tightly delay-constrained applications.

### 3.2 Effect of the Smoothing Period

The next question we target in our experimental evaluation is the setting for the appropriate smoothing period $T$. In particular large values of $T$ would be undesirable as this would result in larger buffers and hence longer set-up times, and in the case of live-feeds longer delays. Figure 5 shows the behavior of $\varepsilon$ and $\varepsilon^{(p)}$ as functions of $T$ for two illustrative streams. For other streams the results we obtained were very similar. In general $\varepsilon$ is very sensitive to small values of $T$, slowly converging to the mean. On the other hand, $\varepsilon^{(p)}$ is essentially independent of $T$ (for $T \geq T_{GoP}$). This means that by choosing the reservation that guarantees the delivery of a $p$ fraction of the groups of frames, we can ensure a low delay, and get the additional benefit of requiring a small buffer, and all of these with just a small overhead over the stream's mean rate.

### 3.3 Effect of the Parameter $p$

As mentioned earlier, $p$ is the parameter of the system that allows it to span the spectrum from being a best-effort system (when $p = 0$) to a system with full reservations (when $p = 1$). Small values of $p$ lead to higher link utilizations, but also to larger values of $\eta$ if we intend to maintain a bound on the number of dropped frames as given by equation (5). Therefore, a full understanding of the choices of $p$ requires the consideration of a specific slack assignment rule and the consideration of multiple streams sharing this capacity. For the following experiments, we setup a trace-driven simulation of the actual streaming process (as described in section 2.4), with arbitrary subsets of streams, and using the maximum rule for the slack assignment.

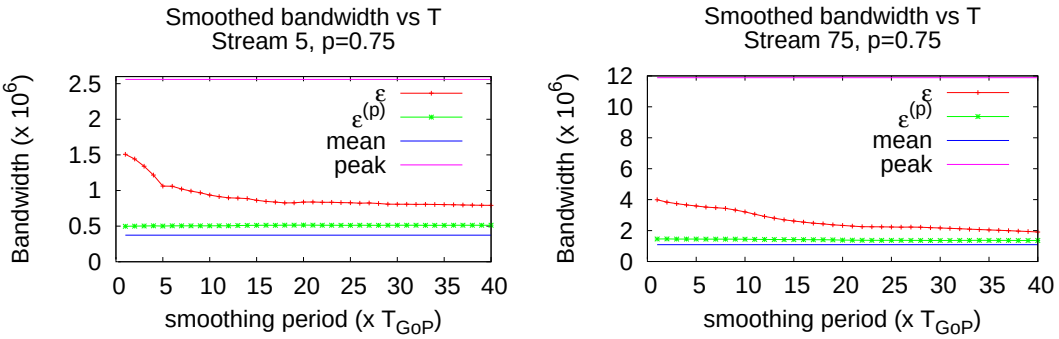



Figure 5. Relation of $\varepsilon$ and $\varepsilon^{(p)}$ to the smoothing period

Figure 6 shows the relationship between the total allocated bandwidth and the utilization as $p$ varies for two different choices of the smoothing period, $T = T_{GoP}$ in part (a) and $T = 5T_{GoP}$ in part (b). Correspondingly, Figure 7 shows the experimental losses (dots) and the Markov bound on the losses (continuous line) as obtained from equation (5). In many cases the losses were zero and thus are not registered on the logarithmic scale of the plot.

The total bandwidth allocated is increasing with increasing values of $p$, *i.e.*, the closer the system gets to a full reservation system, the larger the bandwidth it claims. It is also possible to appreciate that for large values of $p$ the maximum utilization stays below the total allocated bandwidth. This supports the idea of using the $p$-percentile instead of the maximum. It is also worth noting that the differences between the two choices of smoothing periods (one versus five $T_{GoP}$) are very small, as expected from the analysis underlying Figure 5.

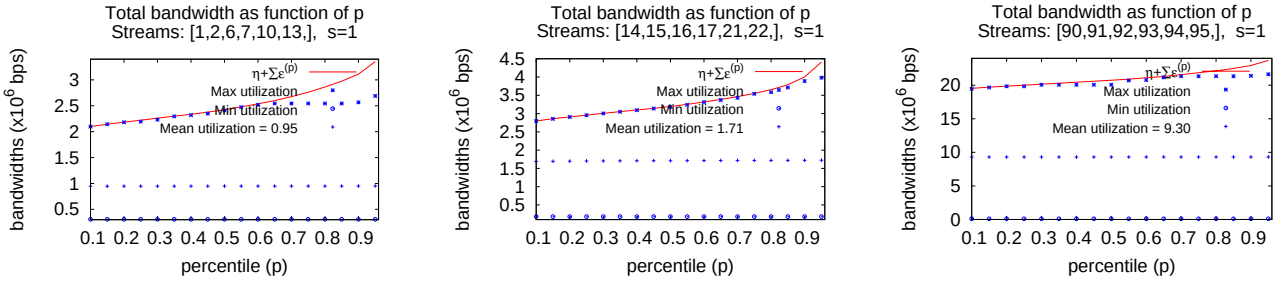
As expected, as we increase the value of the parameter $p$, the actual loss rate of the system decreases. Notice that the Markov bound for the loss rate follows the same trend, but it is always one or more orders of magnitude larger than the actual loss rate of the system. Also, we note that unlike the results for the total allocated bandwidth, the choice of the smoothing period has a significant impact on the loss rates. Having larger smoothing periods (and correspondingly larger buffers) causes a significant reduction in the probability of incurring losses. As a matter of fact for many of the $s = 5$ cases the losses were zero.

These two sets of graphs also underscore the trade-off between bandwidth efficiency and losses. The smaller the value of $p$ (and the closer to a best-effort system), the higher the bandwidth efficiency, but also the larger the losses. Therefore, the optimization problem of minimizing the total bandwidth given QoS constraints on the delay and the losses is the problem of finding the smallest $p$ such that delay and loss constraints are satisfied.

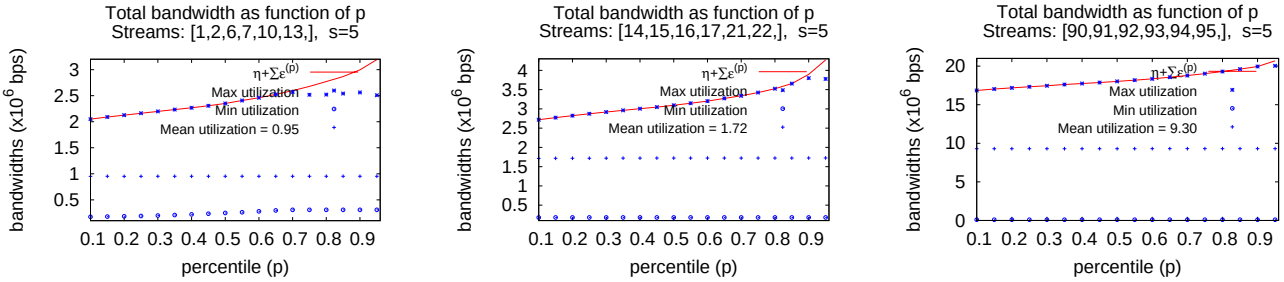
### 3.4 Effect of the Number of Streams

The main objective of our two-tiered bandwidth allocation scheme is to pack as many streams as possible per server, while satisfying the constraints imposed on both the maximum delay and loss rates. Therefore, in this last set of experiments we consider the effect of the multiplexing level ($m$) for a fixed value of the parameter $p$. In Figure 8, for each value of $m$, ten different random combinations of $m$ streams give an equal number of datapoints for both the actual losses and the bound on the losses. We conducted these experiments with a smoothing period $T = T_{GoP}$ and $p = 0.75$. In all cases the Markov bound holds and the system scales for large numbers of streams. For large $m$, and particularly for H.264 streams which exhibit more variability, the losses may go slightly above $1\%$, which could be compensated by defining a larger $\eta$.

Finally, it is interesting to compare our probabilistic two-tiered reservation approach to a deterministic approach in terms of the total bandwidth utilization. As already mentioned, using an effective bandwidth envelope provides the tightest deterministic method to allocate bandwidth to a VBR stream. Figures 9 and 10 show a comparison of the total bandwidth used by our approach to that achievable using an effective bandwidth envelope. Figure 9 shows the actual bandwidth values for increasing levels of concurrency. For each value of $m$ there are $10$ samples using different combinations of streams. In the case of the H.264 traces, there were not enough streams to allow for large enough values of $m$, but the MPEG-4 traces clearly indicate the tendency to achieve bandwidth savings as $m$ increases. Figure 10 shows the tendency to reach close to $50\%$ savings (for the parameters used, $s = 1$, $p = 0.75$) as $m$ gets large enough.

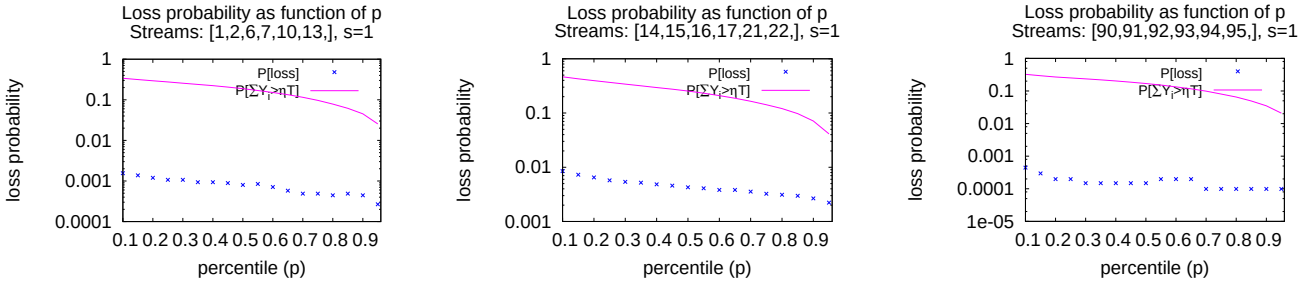



(a) s=1

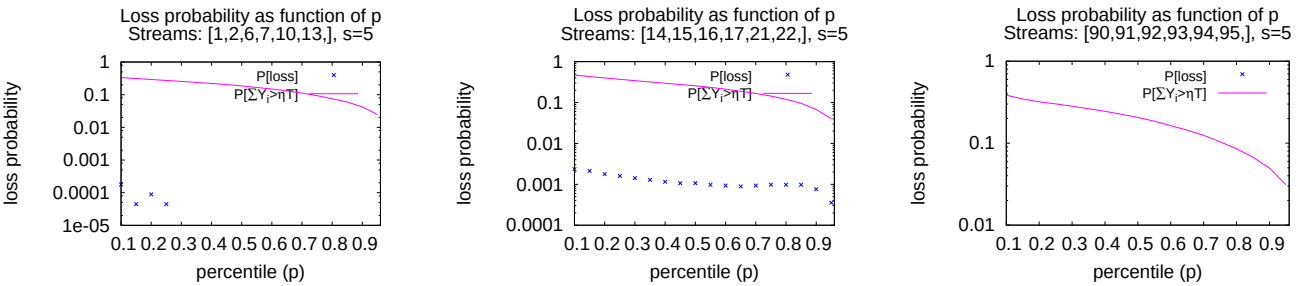



(b) s=5

Figure 6. Bandwidth allocated and utilization as function of $p$

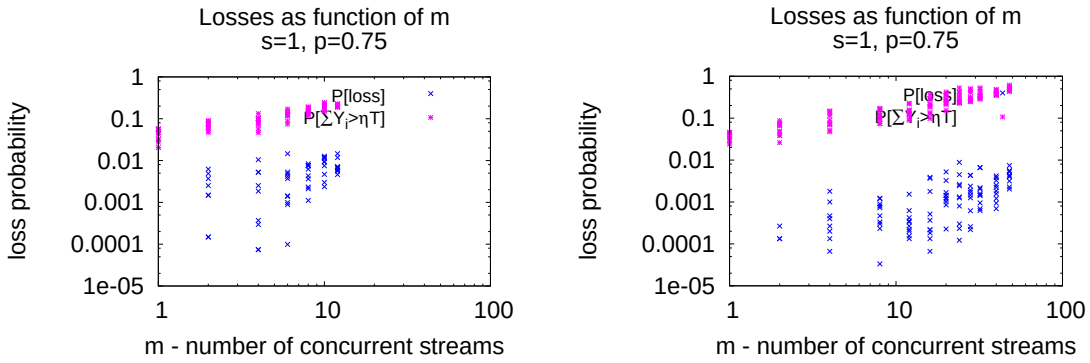



(a) s=1



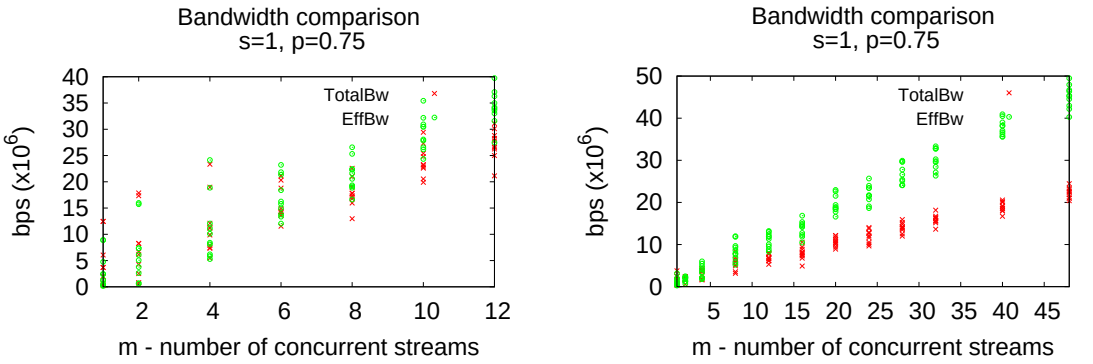

(b) s=5

Figure 7. Losses and the bound on the losses as function of $p$



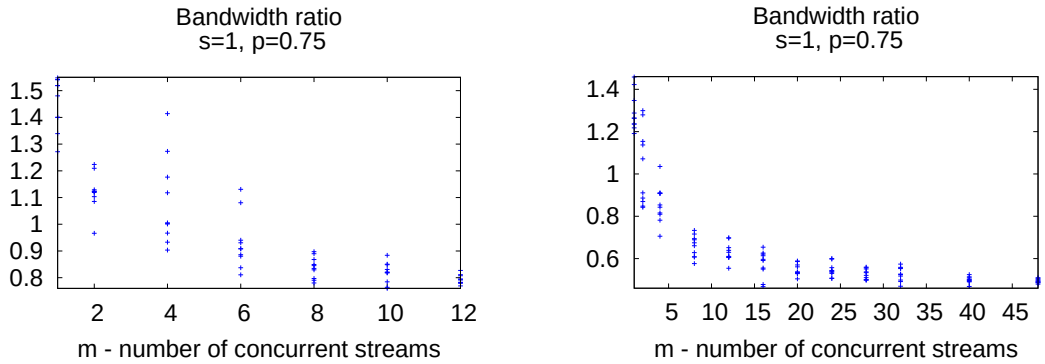

(a) H.264 traces (b) MPEG-4 traces

Figure 8. Losses as function of the number of concurrent streams




(a) H.264 traces (b) MPEG-4 traces

Figure 9. Comparison of effective bandwidth against total bandwidth as function of the number of concurrent streams




(a) H.264 traces (b) MPEG-4 traces

Figure 10. Ratio of total bandwidth to effective bandwidth as function of the number of concurrent streams

## 4. RELATED WORK

There is a long history of research in bandwidth management for video and multimedia applications, a significant portion of which is devoted to the problem of bandwidth allocation and buffer sizing at the client side. For example Sen et al[1] analyzed the problem of determining the minimum buffer size and the corresponding rate and startup latency as required by the client. In doing so, they show there is one such minimum, and present an $O(N \log N)$ algorithm to find it. More recently, Li et al[18] showed how to characterize the stream's envelope using the corresponding parameters of a leaky-bucket, and they give a geometric approach to compute these parameters. In our experimental evaluation, we used the *effective bandwidth* envelope as defined by Mokhtar *et al*.[3] The advantage of this approach is that it gives a tighter bound as compared to leaky-bucket models, but it is also computationally more expensive.

Notice that none of the above-mentioned approaches considers the problem from the server side, and thus none of these approaches is able to capitalize on the economy of scale resulting from the delivery of multiple (potentially large number of) streams from a single server. Moreover, these models are deterministic, lossless and implicitly impose a hard-reservations model.

When looking at the problem from the server side, an important related work is that by Park and van der Schaar,[8] which considers the issue of introducing a brokering system whereby agents bid for the resources they need. In doing so, the problem is modelled as a congestion game which gives the guarantee that there will always be a Nash equilibrium and that the bidding process assures a notion of fairness among players. The principal drawback of this model is that it is supported on an adjustable quality video encoding scheme. From a practical standpoint, on-line video-encoding schemes do not scale as they involve highly CPU-intensive processing. In practice, what large content providers do is to have the videos pre-encoded, possibly at a small discrete number of quality settings, and at delivery time the server's only concern is to deliver the stream under the desired QoS settings.

Su and Wu[9] present another relevant multi-user streaming system that adjusts the encoding rate of each stream while minimizing the mean-square-error and the mean-absolute-difference distortion metrics for each stream. This scheme makes use of the Fine Granularity Scalability (FGS) and Fine Granularity Scalability Temporal (FGST) extensions of the MPEG-4 standard as mechanisms that allow real-time control of the stream's rate. The system then divides the total capacity so that each stream receives the same quality, thus providing for a notion of *quality-fairness*. As before, we argue that adaptive encoding schemes do not scale well to allow service of a large number of concurrent streams.

Yet another server-side approach is that by Krunz and Tripathi[7,19] whereby a technique for allocating VBR streams while minimizing the total bandwidth is proposed. This technique is based on the use of the regularity of the streams' pattern and the maximum sizes of I, P, and B frames. Using these maximum sizes it is possible to define a periodic envelope. By adjusting the temporal shift (phase) between the streams, it is possible to minimize the allocated bandwidth. As with our approach, this technique also incorporates an easy admission control scheme for on-line allocation. The principal drawback of this technique, however, is that it relies on maximum sizes to define the periodic envelope. Due to the very large variations in frame sizes, this envelope is not as tight as that obtained by using the effective bandwidth envelope, or other smoothing approaches, including ours.

There are several techniques[4–6] that find a feasible schedule, *i.e.* an envelope curve, and make use of a reservation mechanism along the path to adjust the reserved bandwidth as dictacted by the schedule. For example, McManus and Ross[4] define periodic constant-rate intervals, whereas Knightly and Zhang[5] generalize the model by allowing arbitrary constant-rate intervals. Lai et al[6] present an allocation scheme for VBR streams that is monotonically decreasing, *i.e.* it dynamically adjust the allocation of a stream, always going downwards. This stands in contrast to other schemes that require adjustments of the allocated bandwidth in the network either increasing or decreasing, where the increasing changes may not be always feasible. Feng and Rexford[2] provide a survey of smoothing techniques and present a detailed evaluation using traces of Motion-JPEG encoded videos.

## 5. CONCLUSION

In this paper, we presented a novel two-tiered approach to the allocation of streaming jobs to (virtual/cloud) servers, in such a way that the allocation be performed in an on-line fashion and that it satisfies the QoS parameters of each allocated stream. To do so, our technique requires knowledge of three static parameters per stream, and needs to keep track of three state variables per server. This makes our proposed approach scalable and very light weight. As a bandwidth

management scheme, our two-tiered allocation strategy spans the full spectrum of design choices, ranging from a best-effort model to exclusive per-stream reservations. Our experimental evaluation confirmed that having an allocation that is shared across a number of streams allows the system to cope quite well with peak traffic demands. In addition, the incurred overhead diminishes as the number of concurrent streams increases. Moreover, the incurred losses can be easily kept within permissible bounds by appropriately provisioning the slack space.

The Markov inequality we used in this paper provides an upper bound on losses in such a way that the bound for a collection of streams is linearly separable. However, this bound is loose. Indeed, our experimental evaluation revealed that losses are typically below this bound by an order of magnitude or more. This opens the possibility of experimenting with many shared slack management techniques, one of which (the maximum rule) was explored in depth during our experimental evaluation. Our current and future work involves the investigation of alternative, and potentially more efficient techniques.

## ACKNOWLEDGMENTS

This work is supported in part by a number of NSF awards, including CISE/CSR Award #0720604, ENG/EFRI Award #0735974, CISE/CNS Award #0524477, CNS/NeTS Award #0520166, CNS/ITR Award #0205294, and CISE/EIA RI Award #0202067. Jorge Londoño is supported in part by the Universidad Pontificia Bolivariana and COLCIENCIAS–Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología "Francisco José de Caldas".

## REFERENCES

[1] Sen, S., Dey, J., Kurose, J., Stankovic, J., and Towsley, D., "Streaming CBR transmission of VBR stored video," in [*Proc. SPIE Symposium on Voice Video and Data Communications*], (1997).

[2] Feng, W. and Rexford, J., "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," in [*Proceedings of INFOCOM '97*], **00**, 58, IEEE Computer Society, Los Alamitos, CA, USA (1997).

[3] Mokhtar, H. M., Pereira, R., and Merabti, M., "An effective bandwidth model for deterministic QoS guarantees of VBR traffic," in [*Proceedings of the Eighth IEEE International Symposium on Computers and Communications ISCC '03*], 1318, IEEE Computer Society, Washington, DC, USA (2003).

[4] McManus, J. and Ross, K., "Video on demand over atm: constant-rate transmission and transport," in [*Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. IEEE INFOCOM '96.*], **3**, 1357–1362 (March 1996).

[5] Knightly, E. W. and Zhang, H., "D-BIND: an accurate traffic model for providing QoS guarantees to VBR traffic," *IEEE/ACM Trans. Netw.* **5**, 219–231 (April 1997).

[6] Lai, H., Lee, J. Y., and Chen, L., "A monotonic-decreasing rate scheduler for variable-bit-rate video streaming," *IEEE Transactions on Circuits and Systems for Video Technology* **15** (February 2005).

[7] Krunz, M., Apostolopoulos, G., and Tripathi, S. K., "Bandwidth allocation and admission control schemes for the distribution of MPEG streams in VOD systems," *International Journal of Parallel and Distributed Systems and Networks* **3**, 108–121 (April 2000).

[8] Park, H. and van der Schaar, M., "Congestion game modeling for brokerage based multimedia resource management," *Packet Video 2007* , 18–25 (November 2007).

[9] Su, G. and Wu, M., "Efficient bandwidth resource allocation for low-delay multiuser video streaming," *IEEE Transactions on Circuits and Systems for Video Technology* **15**, 1124–1137 (September 2005).

[10] Seeling, P., Reisslein, M., and Kulapala, B., "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial," *IEEE Communications Surveys and Tutorials* **6**, 58–78 (2004).

[11] Garrett, M. W. and Willinger, W., "Analysis, modeling and generation of self-similar VBR video traffic," *SIGCOMM Computer Communications Review* **24**, 269–280 (October 1994).

[12] Krunz, M. and Tripathi, S. K., "On the characterization of VBR MPEG streams," in [*Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS '97*], 192–202, ACM, New York, NY, USA (1997).

[13] Fitzek, F. H. P. and Reisslein, M., "MPEG-4 and H.263 video traces for network performance evaluation," *IEEE Network* , 40–54 (2001).

[14] Fitzek, F. H. P. and Reisslein, M., "MPEG-4 and H.263 video traces for network performance evaluation," tech. rep., Technical University Berlin (2000).

[15] Seeling, P., Fitzek, F. H., and Reisslein, M., [*Video Traces for Network Performance Evaluation*], Springer (2007).

[16] "MPEG-4 and H.263 video traces for network performance evaluation." http://www.tkn.tu-berlin.de/research/trace/trace.html (2008).

[17] "Video traces for network performance evaluation." http://trace.eas.asu.edu/tracemain.html.

[18] Li, P., Lin, W., Rahardja, S., Lin, X., Yang, X., and Li, Z., "Geometrically determining the leaky bucket parameters for video streaming over constant bit-rate channels," in [*Proc. of ICASSP04*], **20**, 193–204 (February 2005).

[19] Krunz, M. and Tripathi, S. K., "Exploiting the temporal structure of MPEG video for the reduction of bandwidth requirements," in [*Proceedings of INFOCOM '97*], 67, IEEE Computer Society, Washington, DC, USA (April 1997).