Delimiting the range of effectiveness of scalable on-demand streaming

Haonan Tan[*,a], Derek L. Eager[**,b], Mary K. Vernon[***,a]

[a]*Computer Sciences Department, University of Wisconsin – Madison, USA*
[b]*Department of Computer Science, University of Saskatchewan, Canada*

Abstract

Previous analyses of scalable streaming protocols for delivery of stored multimedia have largely focused on how the server bandwidth required for full-file delivery scales as the client request rate increases or as the start-up delay is decreased. This previous work leaves unanswered three questions that can substantively impact the desirability of using these protocols in some application domains, namely:

(1) Are simpler scalable download protocols preferable to scalable streaming protocols in contexts where substantial start-up delays can be tolerated?

(2) If client requests are for (perhaps arbitrary) intervals of the media file rather than the full file, are there conditions under which streaming is not scalable (i.e., no streaming protocol can achieve sub-linear scaling of required server bandwidth with request rate)?

(3) For systems delivering a large collection of objects with a heavy-tailed distribution of file popularity, can scalable streaming substantially reduce the total server bandwidth requirement, or will this requirement be largely dominated by the required bandwidth for relatively cold objects?

This paper addresses these questions primarily through the development of tight lower bounds on required server bandwidth, under the assumption of Poisson, independent client requests. Implications for other arrival processes are also discussed. Previous work and results presented in this paper suggest that these bounds can be approached by implementable policies. With respect to the first question, the results show that scalable streaming protocols require significantly lower server bandwidth in comparison to download protocols for start-up delays up to a large fraction of the media playback duration. For the second question, we find that in the worst-case interval access model, the minimum required server bandwidth, assuming immediate service to each client, scales as the square root of the request rate. Finally, for the third question, we show that scalable streaming can provide a factor of *log K* improvement in the total minimum required server bandwidth for immediate service, as the number of objects *K* is scaled, for systems with fixed minimum object request popularity.

[*] Email address: haonan@cs.wisc.edu. Research of this author was partially supported by the National Science Foundation (NSF) under grant ANI-0117810.

[**] Email address: eager@cs.usask.ca. Research of this author was partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant OGP-0000264.

[***] Email address: vernon@cs.wisc.edu. Research of this author was partially supported by the National Science Foundation (NSF) under grants CCR-9975044, ANI-0117810 and EIA-0127857.

To appear in *Proc. IFIP WG 7.3 Int'l. Symp. on Computer Performance Modeling, Measurement and Evaluation (Performance 2002)*, Rome, Sept. 2002.

1. Introduction

Scalable streaming protocols (e.g, [4, 5, 8-19, 23, 24]) have been proposed that can efficiently deliver multimedia files with server and network bandwidth that scales much less than linearly in the request rate. Although scalable streaming offers the promise of efficient, low-delay service to large numbers of clients, previous work does not fully address three questions that could have significant impact on the desirability of this technology in some application domains.

First, if clients request the full file and can tolerate relatively high start-up delays (say, approaching the file playback duration), it is unknown how the scalable streaming protocols compare to scalable download protocols (e.g., [1,3,6,20]) with respect to required server bandwidth[1]. By *download*, we mean protocols in which clients receive all of the media data prior to beginning playback, in contrast to *streaming* in which clients can begin playback while concurrently receiving subsequent media data. Since, scalable download protocols do not need to ensure that data reaches the client in time to avoid jitter after the client has begun playback, such protocols can be simpler to implement than scalable streaming protocols.

Second, in some application domains clients frequently request intervals of the media file rather than the full file. For example, when clients access educational videos, they may request various (small) sections of the video that they need to review [2]. Since scalable streaming protocols are based on exploiting the commonality of the data needed to serve approximately concurrent requests for the same file, it is not clear how effective scalable streaming will be for interval requests. Two recent measurement studies of media workloads that include many requests for relatively short intervals [2,7] concluded that scalable streaming has the potential to significantly reduce server bandwidth for the measured workload. Parallel work to this study [18] shows that for a particular access type (namely, fixed size intervals each beginning at a random starting point within the file), the minimum required server bandwidth for immediate service scales with the square root of the request rate. This is a significant result, as all previous analytic studies had assumed full-file access. In this paper we extend this work by considering other access patterns and by determining the worst-case scaling of the minimum required server bandwidth with request rate.

Finally, scalable streaming offers substantial improvements over simple unicast delivery only for sufficiently popular content. An open question is how effective scalable streaming protocols are for servers with many media files and a heavy-tailed distribution of media file popularities.

This paper investigates these questions by developing tight lower bounds on required server bandwidth for the relevant protocols or class of protocols, under the assumption of Poisson, independent client requests. Implications for other arrival processes are also discussed. For the scalable streaming protocols that provide immediate service, the required server bandwidth is the average bandwidth that is used if the server has enough bandwidth to provide every client with immediate service (i.e., there is no client queueing or balking). Recent work [13, 22] has found that scalable streaming servers achieve negligible client waiting time when configured with finite bandwidth equal to a small percentage more than this required bandwidth. This is to be expected because the total bandwidth used to provide immediate access to a set of independently requested

[1] By *required server bandwidth*, we mean the average server bandwidth used by a media delivery protocol. Typically of interest in this paper is a *tight* lower bound on the required server bandwidth used by any protocol in some class (such as a particular class of download protocols, for example, or streaming protocols that provide immediate service), which we term the *minimum required server bandwidth* for that class of protocols.

files will have lower coefficient of variation over time than the bandwidth used to provide access to one of the files.

Key results of the analysis include the following:

- Scalable streaming protocols can yield significant benefits in comparison to download protocols for start-up delays that are up to a large fraction of the media playback duration. Scalable streaming can also yield (smaller) benefits for larger start-up delays; i.e., the potential benefit smoothly decreases, but stays strictly positive, as the start-up delay increases.
- The *worst-case* scaling of the minimum required server bandwidth for immediate service is as $\sqrt{\frac{2}{\pi}N}$, where N is the normalized request rate for the file (i.e., N is the average number of client arrivals within a period of time equal to the full-file media playback duration).
- In the context of large-scale media delivery systems with a heavy tailed (e.g., Zipf) distribution of file popularities and K files in total, assuming a fixed, arbitrarily small minimum file access frequency as the system scales, scalable streaming can provide a log K improvement in the scaling of the total minimum required server bandwidth for immediate service, in comparison to unicast delivery.

The remainder of the paper is organized as follows. Section 2 reviews the hierarchical stream merging protocol and previously developed server bandwidth bounds for scalable streaming. Section 3 develops bounds on required server bandwidth for download protocols, and applies these to compare the performance of download and streaming under conditions of substantial client start-up delays. Section 4 examines the scalability of streaming in the context of interval rather than full-file access. The effectiveness of scalable streaming for large media servers with many relatively cold objects is considered in Section 5. Section 6 concludes the paper.

2. Background

2.1. Hierarchical stream merging

Previously proposed scalable streaming methods include *periodic broadcast* protocols (e.g., [5,15,16,19]), *patching* (e.g., [8,9,14,17]), and *hierarchical stream merging (HSM)* [4,10-13]. The key trade-offs among these protocols are discussed in [12,19,22]. This section briefly reviews the operation of HSM, a streaming method for which we present performance results later in the paper.

HSM protocols provide *immediate service* to each request and, as described in Section 2.2, have required server bandwidth for full-file accesses that grows only *logarithmically* with the request rate. An example illustrating the basic operation of HSM is provided in Figure 1. In the figure, clients A through D request the same full media file at times T1, T2, T3, and T4, respectively. Clients B and D simultaneously listen to their own stream, and that of clients A and C, respectively. Once clients B and D have "caught up" to clients A and C (i.e., with respect to the file data that they have received), their own individual streams can be terminated. The "merged" group of clients C, D then listens to the stream for merged clients A and B, as well as its own stream, and all four clients eventually merge. A variety of rules can be defined for which additional stream, if any, a client listens to in order to hierarchically merge with other clients. In the simple "closest target" policy, a client listens to the closest earlier stream that is still active [12, 13]. A number of other policies have been defined that yield very similar performance [4,10,13].

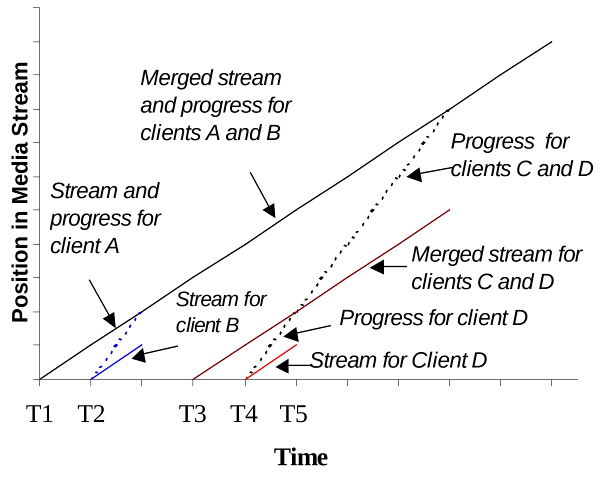

Position in Media Stream
Merged stream and progress for clients A and B
Progress for clients C and D
Stream and progress for client A
Merged stream for clients C and D
Stream for client B
Progress for client D
Stream for Client D
T1
T2
T3
T4
T5
Time

Figure 1: HSM Example

Table 1: Notation

Symbol	Definition
λ	Average request rate (for one file, or total for all files, depending on context)
K	Number of files
T	File playback duration
N	Average number of requests during period of length T ($N = \lambda T$)
d	Client start-up delay
D	Client delay expressed as a fraction of the file play duration ($D = d/T$)
B_{min}	Minimum required server bandwidth

For clients with finite buffer space, HSM protocols can be easily modified so that merges that would otherwise overrun a client's buffer are simply not scheduled [11, 12]. The application of the closest target HSM protocol for interval requests is discussed in [12]. A prototype [19] that implements this protocol currently operates seamlessly in production mode for the Eteach server that has highly interactive clients (see Appendix A). An alternative protocol applicable to interval requests is proposed in [18], and has the advantage that each client need only join one multicast stream.

Figure 1 and the description of the HSM protocol above assume that each client receives immediate service, but HSM can be combined with a batching policy [11]. In this case service to a request may be delayed, during which time any other requests for the same media object are batched together. The stream for the batch of clients then merges hierarchically with earlier streams.

2.2. Tight lower bounds on required server bandwidth for scalable streaming

In previous work [12], parallel work [18], and in this paper, the minimum required server bandwidth for a class of protocols that initiate streams in response to client requests is derived assuming Poisson client request arrivals. Poisson arrivals have been observed for the requests to view videos in a production video server on the U.C. Berkeley campus [2]. For interarrival times with higher variance, such as heavy-tailed interarrival times (as have been observed for the interactive request arrivals in the eTeach server [2]), the Poisson arrival assumption for the same mean arrival rate leads to conservative (i.e., *pessimistic*) estimates of the required bandwidth for scalable streaming protocols that provide immediate service [11]. This is because more bursty arrivals imply a shorter time to merge streams, on average. We note that approximate bandwidth estimates are generally more useful if they are conservative, particularly since an actual system will generally be configured with somewhat higher than the estimated required bandwidth even in the case of accurate estimates. Moreover, the coefficient of variation in the observed heavy-tailed distribution of interarrival times for interactive requests in the eTeach system is between 1 and 2 (i.e., not much higher than that for Poisson arrivals). Thus, one might expect that the Poisson assumption will capture the body of the interarrival time distribution reasonably well and thus predict required server bandwidth that is reasonably accurate for such client workloads.

The minimum required server bandwidth formulas derived in this paper further assume that clients have sufficient buffer capacity to receive all media data that is delivered in advance of playout and that clients can receive arbitrarily many streams concurrently. For protocols that provide immediate service, previous work [12] and parallel work [18] show that the minimum required server bandwidth derived in this way can be nearly achieved (i.e., within a small constant factor) with practical protocols that assume that clients can receive two or fewer streams concurrently, for full-file requests or interval requests, even when clients have relatively small buffer capacity [11] such as 15% of the file. For periodic broadcast protocols with unconstrained client buffer capacity, similarly modest client data rate is sufficient to nearly achieve the minimum required server bandwidth for this class of protocols [19].

Of interest first is the minimum required server bandwidth for protocols that provide immediate service to each client, for full-file delivery of a single media file, assuming Poisson client requests and sufficient buffer capacity to receive all media data that is delivered in advance of playout. Using the notation in Table 1, a *tight* lower bound on the required server bandwidth for such protocols, measured in units of the streaming rate, is given by [12]:

$$B_{min}^{d=0} = \int_0^T \frac{\mathrm{d}x}{x+\frac{1}{\lambda}} = \ln(T\lambda+1) = \ln(N+1)\,. \tag{1}$$

This bound is derived by considering a small portion of the file at an arbitrary time offset *x*. For a client request that arrives at time *t*, this portion of the file must be delivered no later than time *t*+*x* if the client starts play out (from the beginning of the media file) right away and if the data is to arrive on time. If it is multicast as late as possible, i.e., at time *t*+*x*, then (at best) those clients that request the file between time *t* and *t*+*x*, can receive the same multicast. Since the average time from *t*+*x* until the next request for the file is 1/λ, the minimum frequency of multicasts of the portion at time offset *x* is 1/(*x*+1/*λ*), which yields the bound.

The lower bound in equation (1) can be modified to obtain the minimum required server bandwidth for any streaming system that has a maximum start-up delay *d*>0, by adding *d* to the minimum time between multicasts of each portion of the media file [21]:

$$B_{min} = \int_0^T \frac{\mathrm{d}x}{x+d+\frac{1}{\lambda}} = \ln\left(\frac{T}{d+1/\lambda}+1\right) = \ln\left(\frac{N}{ND+1}+1\right). \tag{2}$$

3. Server bandwidth comparisons: scalable download versus scalable streaming

With respect to streaming delivery, most attention has been focused on the server bandwidth required for either very short startup delay or for immediate service. Suppose, however, that clients can tolerate (or have paid for a lower-cost service that has) a relatively long maximum startup delay (*d*), equal to a significant fraction of the playback duration (for example, fifteen minutes for a half hour TV show), and assume that the bandwidth to the client is such that the object can be downloaded within the client delay constraint. Again, by download, we mean protocols in which clients receive all of the media data prior to beginning playback, in contrast to streaming in which clients can begin playback while concurrently receiving subsequent media data. In the case of download, the server could employ (for example) the simple scalable download protocol that uses a single server stream to cyclically multicast the media file data.[2] Clients begin listening to the multicast at the time of their request, and stop listening when the entire file has been downloaded. A basic question is whether there is any substantive advantage to be gained from use of scalable streaming protocols in such a context.

One might think that there are conditions under which scalable download is inherently more efficient, since there are no constraints on the order in which the data is delivered. On the other hand, for any given start-up delay constraint, one possible streaming protocol is to use the optimal download protocol. Thus, the minimum required server bandwidth for scalable streaming is no greater than the bandwidth required for download.

In this section we compare the minimum required server bandwidth of scalable streaming (i.e., the best that can be achieved by any scalable streaming protocol), to the minimum required server bandwidth of scalable download (i.e., the best that can be achieved by any scalable download protocol). For comparison we also derive the required server bandwidth for two simple scalable download protocols: "in-order download" and "cyclic download". With in-order download, clients receive media data in-order from a single full-file transmission, and wait until the entire file has been received before beginning playback. With cyclic download, a single stream is used to deliver the file (at rate T/d in units of the media play rate), whenever at least one client is listening to the multicast.

The analysis below uses the notation in Table 1 and assumes request arrivals are Poisson at rate λ, each request is for the full media file, clients can receive the media data at an arbitrarily high rate, and, as required in download systems, clients have buffer space for the requested file. For reasons explained for streaming in Section 2.2, these assumptions yield results that are widely applicable. In particular, Poisson arrivals have been observed [2], and can be expected in many environments, for requests to receive a given (full) file. Furthermore, as with scalable streaming, scalable download protocols can often achieve close to the minimum required server bandwidth for downloads with only modest client data rate, as is shown in the results below. For example, for start-up delay greater than one half the total playback duration (i.e., $d > 0.5$), cyclic download yields close to the minimum download bandwidth requirement, and yet requires client data rate less than twice the media play rate.

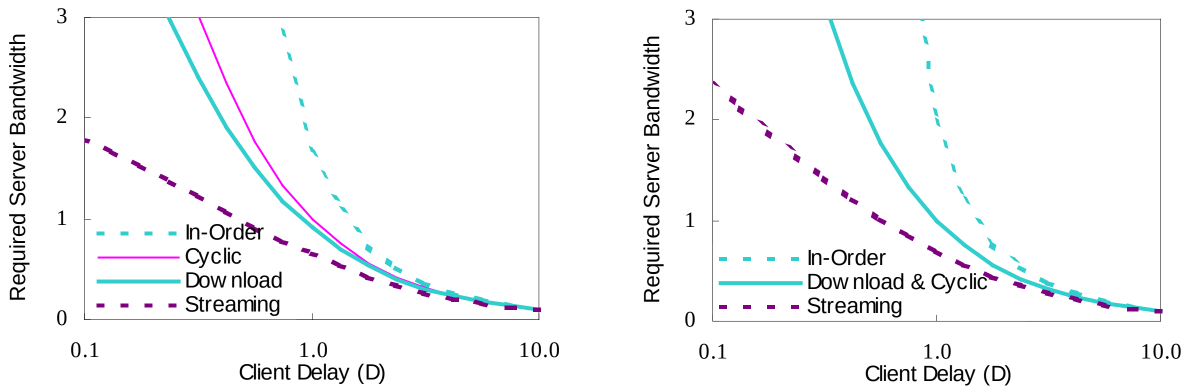
For streaming, a tight lower bound for arbitrary start-up delay was given in equation (2).

For in-order download and transmission rate to each client, in units of the media play rate, equal to r ($r \geq T/d$), the minimum frequency with which full file multicasts must begin is $1/(d - T/r + 1/\lambda)$. The required server bandwidth is therefore given by

$$B_{min}^{in-order\ download} = \frac{T}{d - T/r + 1/\lambda} = \frac{1}{D - 1/r + 1/N} . \qquad (3)$$

For the cyclic download protocol, the server streams the data at rate T/d and an arriving client request finds an active transmission of the file if the previous request occurred less than d units of time earlier. The probability that the request arrival finds the active transmission, by PASTA, is the fraction of time that the stream is active. Thus, the required server bandwidth is given by

[2] In addition to simplicity, this approach has the advantage of enabling an elegant approach to packet loss recovery [20][6].



(a) Client Request Rate $N = 10$ (b) Client Request Rate $N = 1000$

Figure 2: Required Server Bandwidth for Download Compared to Streaming

$$B_{min}^{cyclic\ download} = \frac{T}{d}\left(1-e^{-\lambda d}\right) = \frac{1}{D}\left(1-e^{-ND}\right). \tag{4}$$

Consider now the minimum required server bandwidth for *any* download protocol. The minimum frequency with which each portion of the media file must be multicast, so as to ensure that each client receives all data within time d of its request is $1/(d + 1/\lambda)$. Thus, a lower bound on the required server bandwidth for a file of duration T, in units of the play rate, is given by

$$B_{min}^{download} = \frac{T}{d+1/\lambda} = \frac{1}{D+1/N}. \tag{5}$$

Note that achieving this bound requires that each client be able to receive the entire media file at the instant that is d units after the client request that triggers the transmission.

Figure 2(a) and (b) show the above minimum required server bandwidths for streaming and download, as a function of the normalized start-up delay $D = d/T$, for request rates N equal to 10 and 1000, respectively. As noted above, both of these bounds assume unlimited client bandwidth, but each bound can be nearly achieved by protocols that assume modest client data rate. The figure also shows the required server bandwidth for cyclic download and for the in-order download protocol with $r = 2$. The results indicate that scalable streaming protocols can yield significant benefit, even for start-up delays that are a large fraction of the media playback duration (i.e., D close to 1).

We also note that, if client request interarrival times have higher variance than the Poisson, the gap between the minimum bandwidth for streaming and the minimum bandwidth for download may be somewhat smaller than in Figure 2. To see this, consider the case where $D = 1$ and the arrival process alternates between (1) a very large burst of requests that have negligible interarrival times, and (2) a very long interarrival time. In this (unrealistic) case, the required server bandwidth for streaming is essentially the same as the server bandwidth for cyclic download, which is approximately one stream at the media play rate per cycle. Quantitative comparisons of streaming and download systems for high start-up delay and observed arrival processes with higher variance than Poisson arrivals is left for future work.

4. Interval access

In some environments (e.g., educational video delivery [2]), most client requests are for intervals of media data, rather than for full files, as illustrated in Appendix A. Since there is less commonality in the data that the clients are requesting, and since the play durations of the requested intervals may be quite short, scalable streaming can be expected to be less effective. On the other hand, two recent streaming workload studies [2,7] found evidence that scalable streaming could significantly reduce bandwidth usage in the measured environments, each of which included substantial interval access. For example, simulations of the HSM protocol for the client requests to the most popular files on the eTeach server are reported in [2]. Those results show server bandwidth for HSM between 40% – 60% lower than for unicast streaming, although 90% of the requests are for fewer than three minutes of the video and the total request rate for each simulated period is between 10 and 70 requests/hour. An interesting question is whether such significant bandwidth savings could be expected in systems with Poisson arrivals and independent interval requests, assuming similar arrival rate and interval access pattern to the measured interval requests simulated in [2]. That is, can significant bandwidth savings occur when there is no special temporal locality in the relatively short intervals, and the request rate is relatively low?

An analytic study is provided in parallel work in [18] for the case of one particular access type, in which clients request an interval of fixed size (with wrap around) beginning from a random (uniformly distributed) starting point within the file. This section provides the minimum required server bandwidth for immediate service for various other interval access request models, and applies this analysis to address two fundamental questions:

- What is the worst case scaling with request rate of the minimum required server bandwidth, for independent, Poisson client requests accessing arbitrary intervals of media file data?
- Under what types of interval access patterns would one expect the worst case scalability behavior, or alternatively the logarithmic growth established for full-file access?

Four models of media interval accesses are considered below. First, the requested interval is from the beginning of the file to a random (uniformly distributed) ending point. This pattern was observed in the eTeach system (see Appendix A), and might also represent client browsing behavior in other systems. Second, the requested interval has a random (uniformly distributed) starting point, and continues to the end of the media file. The third type has both a random start and a random end point for each interval, motivated by the observed requests in Figure 7(a) of Appendix A. The fourth, quite general model of interval accesses, is where the media file has an arbitrary number of markers, and requests are for intervals that begin at each marker with a specified probability distribution, with the end of the interval either the end of the file or specified by another probability distribution that is dependent on the starting point. This fourth model is motivated by the observed requests shown in Figure 7(b).

Results for the four types of accesses yield insight into the properties of the access type that have a principal impact on the required server bandwidth. As might be expected, the first and fourth models result in significantly better scaling of minimum server bandwidth with client request rate, as there is more opportunity for stream merging to occur reasonably soon after a request arrives. Further, as will be shown in Section 4.5, the second model defines the interval access type that has the worst case scaling of the minimum required server bandwidth with request arrival rate, for the case of Poisson arrivals and independent interval requests.

Each of the four access models, in order, is analyzed in Sections 4.1- 4.4 below, respectively. The analysis of the fourth access model is used at the end of Section 4.4 to address the question posed above about whether the bandwidth savings projected from simulations of actual client traces are obtained for independent Poisson requests with a similar access pattern. Section 4.5 addresses the fundamental questions posed above regarding the worst case scaling behavior for interval requests.

4.1. Random ending point

When the requested portion of the media file is from the beginning of the file to a uniformly distributed ending point, the derivation of the minimum required server bandwidth is very similar to that used for equation (1). Consider an infinitesimally small file segment with offset *x* (in terms of playback time) relative to the beginning of the file. Suppose that at time *t*, a client request arrives and the requested interval includes that specific file segment. This segment must be received no later than time *t+x*. If no earlier transmissions of the segment have been scheduled, and the segment is multicast at the latest possible time *t+x*, all other client requests that arrive between time *t* and *t+x* can receive this multicast. For Poisson arrivals of requests at rate λ with uniformly distributed ending points of the requested intervals, the arrival rate of requests that need access to the segment at offset *x* is $\lambda(T-x)/T$. Thus, the average time from *t+x* until the arrival of the next request that would need to trigger another transmission of the segment is $T/[(T-x)\lambda]$, and the expected time interval between two consecutive transmissions (the "retransmission interval") of the file segment at offset *x* must be at most $x+T/[(T-x)\lambda]$. If the retransmission interval is denoted as a random variable *Z*, using the notation of conditional expectation, this can be written as $E[Z|x]=x+T/[(T-x)\lambda]$. Therefore, a lower bound on the required server bandwidth for this access pattern is

$$B_{min}^{random\ \ end} = \int_0^T \frac{\mathrm{d}x}{E[Z\,|\,x]} = \int_0^T \frac{\mathrm{d}x}{x + \dfrac{T}{(T-x)\lambda}} = \frac{N}{\sqrt{N^2+4N}} \ln\left(\frac{N+\sqrt{N^2+4N}}{2} + 1 \right). \qquad (6)$$

Note that as with full-file playback, for large *N* the minimum required server bandwidth scales with the logarithm of the client request rate (*N*).

4.2. Random starting point

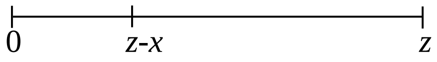
A seemingly symmetric but actually more intricate situation is where the requested interval starts at a random point (uniformly distributed) in the media file and continues to the end of the file. With the previous access type, *Z* can never be smaller than *x* because every access starts at the beginning of the media file. When the start point is random, however, *Z* can be arbitrarily smaller than *x*. Assuming Poisson arrivals, the following result holds:

$$E[Z\,|\,x] = \int_0^\infty z \exp(-\bar{\lambda}(x) z) \cdot \tilde{\lambda}(x) \mathrm{d}z\ , \qquad (7)$$

where $\bar{\lambda}(x)$ is the average arrival rate of requests (computed over a period of length *z*) that would result in a retransmission interval of less than *z*, and $\tilde{\lambda}(x)dz$ is the probability of a request arrival occurring at the specific time that would result in a retransmission interval of precisely *z* (should there have been no previous request arrivals that result in a shorter retransmission interval). Note that

$\exp(-\bar{\lambda}(x)z)$ gives the probability of no request arrivals that would cause a shorter retransmission interval than *z*, even when the arrival rate is time-varying over this interval, as long as the request arrival process is Poisson over this interval (which is seen to be the case in the scenarios considered below).

To illustrate how $\bar{\lambda}(x)$ and $\tilde{\lambda}(x)$ are determined, consider the simpler case of full-file access. In this case, the retransmission interval *Z* is never shorter than *x*. Consider an arbitrary $z > x$ and the timeline show below:



Suppose that at time 0 the small segment at offset *x* is transmitted. In order for its first retransmission to occur at time *z*, two requirements have to be met. First, there must be no "early-triggering" requests, i.e., no requests coming in after time 0 that will need this segment earlier than time *z*. Second, there must be a request for which the segment must be delivered no later than precisely time *z* if it is to be received on time. For the first requirement, it is clear that between time 0 and *z-x*, any incoming requests would be early-triggering. The arrival rate of such requests is λ on (0, *z-x*). On the other hand, no requests arriving between time *z-x* and z are early-triggering, so the arrival rate of such requests is 0 on (*z-x*, *z*). This leads to the expression for the average arrival rate

$$\bar{\lambda}(x) = \frac{\lambda \cdot (z-x) + 0 \cdot x}{z} = \frac{z-x}{z}\lambda, \tag{8}$$

and thus

$$\exp(-\bar{\lambda}(x)z) = \exp[-\lambda(z-x)]. \tag{9}$$

In this simple case, the above result could have been obtained by simply noting that early-triggering requests occur at rate λ during an interval of length *z-x*, but our alternative method is useful for more complex cases.

Finally, note that any request that arrives at time *z-x* causes a retransmission at time *z* (assuming no earlier request arrivals), and thus

$$\tilde{\lambda}(x) = \lambda \ , \tag{10}$$

yielding, for full-file access,

$$E[Z \mid x] = \int_x^{\infty} z \exp(-\lambda(z-x)) \cdot \lambda \mathrm{d}z = x + \frac{1}{\lambda}, \tag{11}$$

which agrees with equation (1).

Now consider the case where the requested portion of the media file is from the beginning of the file to a uniformly distributed ending point. It is still true that early-triggering requests can only arrive within (0, *z-x*). The rate differs, however, since a request does not require a retransmission if its requested interval ends before *x*, and thus the average arrival rate of early-triggering requests over (0, *z-x*) is λ*(T-x)/T* as derived previously, yielding

$$\bar{\lambda}(x) = \frac{(T-x)\lambda / T \cdot (z-x) + 0 \cdot x}{z} = \frac{(z-x)(T-x)}{zT}\lambda. \tag{12}$$

Since only arrivals whose requested interval does not end before position *x* can cause a retransmission,

$$\tilde{\lambda}(x) = \frac{T-x}{T}\lambda\,, \tag{13}$$

and equation (7) becomes

$$E[Z \mid x] = \int_x^\infty z\exp(-\lambda\frac{(z-x)(T-x)}{T})\cdot\frac{T-x}{T}\lambda dz = x + \frac{1}{\tilde{\lambda}} = x + \frac{T}{(T-x)\lambda}, \tag{14}$$

which agrees with equation (6).

In the case where each requested interval starts at a random point (uniformly distributed) in the media file and continues to the end, it is necessary to consider both $z > x$ and $z < x$.

For $z > x$, every request that arrives within (0, *z*-*x*) and whose requested interval starts before *x* is early-triggering since every requested interval continues to the end of the file. Thus, the arrival rate of early-triggering requests over (0, *z*-*x*) is $\lambda x/T$. However, the rate is no longer zero, and not even constant, over (*z*-*x*, *z*). A request that arrives *y* units of time before time *z* ($0 < y < x$) is early-triggering if and only if its start point offset is between *x*-*y* and *x*. Therefore, the arrival rate at time *z*-*y* of early-triggering requests is $\lambda(x-y)/T$. Consequently, the average arrival rate of early-triggering requests over the interval *z*, for the case of $z > x$, is given by

$$\bar{\lambda}_{z>x}(x) = \frac{x\lambda(z-x)/T + \int_0^x \frac{x-(x-y)}{T}\lambda dy}{z} = \frac{x}{Tz}(z-\frac{x}{2})\lambda\,. \tag{15}$$

A request can cause a retransmission at time *z* only if it arrives *y* units of time before time *z* ($0 < y < x$), and the starting point of the requested interval is *x*-*y*, implying that

$$\tilde{\lambda}_{z>x}(x) = \int_0^x \frac{\lambda dy}{T} = \frac{x}{T}\lambda\,. \tag{16}$$

For $z < x$, consider a request that arrives *y* units of time before time *z* ($0 < y < z$). It is early-triggering if and only if the starting point of the requested interval is between *x*-*y* and *x*, implying that

$$\bar{\lambda}_{z<x}(x) = \frac{\int_0^z \frac{x-(x-y)}{T}\lambda dy}{z} = \frac{z}{2T}\lambda\,. \tag{17}$$

A retransmission can be caused by an arrival at any time between 0 and *z* (depending on the starting point of the requested interval), and thus

$$\tilde{\lambda}_{z<x}(x) = \int_0^z \frac{\lambda dy}{T} = \frac{z}{T}\lambda\,. \tag{18}$$

From equations (7) and (15)-(18),

$$\begin{aligned} E[Z \mid x] &= \int_0^x z\exp(-\frac{z^2}{2T}\lambda)\cdot\frac{z}{T}\lambda dz + \int_x^\infty z\exp[-\frac{x}{T}(z-\frac{x}{2})\lambda]\cdot\frac{x}{T}\lambda dz \\ &= \sqrt{\frac{2\pi T}{\lambda}}\Phi(\sqrt{\frac{\lambda}{T}}x) + \frac{T}{x\lambda}\exp(-\frac{\lambda}{2T}x^2), \end{aligned} \tag{19}$$

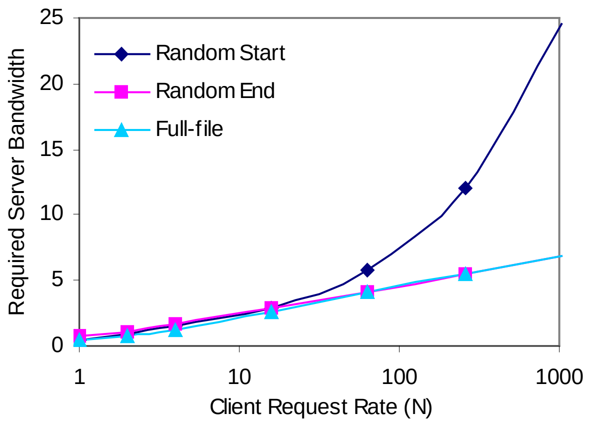


Figure 3: Minimum Required Server Bandwidth with Interval Accesses Vs. Full-file Accesses

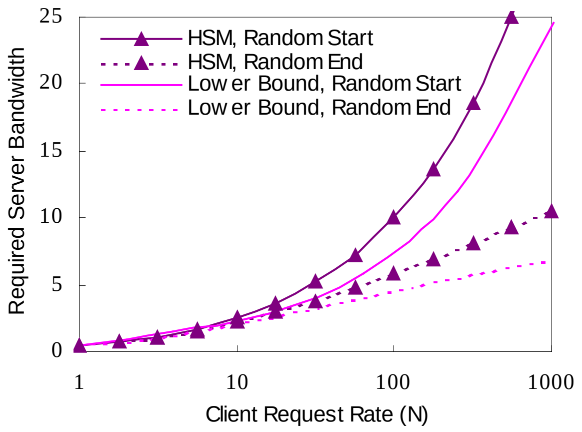


Figure 4: Required Server Bandwidth for HSM with Interval Accesses

where $\Phi(t) = \int_0^t \frac{1}{\sqrt{2\pi}} \exp(-\frac{s^2}{2}) ds$. Hence,

$$B_{min}^{random\ \ start} = \int_0^T \frac{dx}{E[Z \mid x]}, \tag{20}$$

and it is shown in Appendix B that

$$\lim_{N \to \infty} \frac{B_{min}^{random\ \ start}}{\sqrt{\frac{2}{\pi} N}} = 1. \tag{21}$$

This result is similar to that obtained in [18] for the access type with fixed size intervals that start at a random point in the file and wrap-around.

Figure 3 compares the minimum required server bandwidth with interval access, of either the random start or random end type, to that with full file access. For the random start access type, results were obtained by numerical integration. Note that there is very little difference between the minimum required server bandwidth with the random end type of access, and with full-file access.

As illustrated in the figure (and shown in the analytic results derived above), even though the average request length, and the average amount of "overlap" between two different requests, is the same in both the random start and random end cases, the bandwidth requirement for random start is substantially higher. The explanation for this arises from the length of time during which a client is able to receive some multicast transmission of a required portion of the media file, in time for play out. This time is very short at the beginning of a requested interval, and, for this reason, a high intensity of other concurrent client requests for the beginning of the interval (as can occur when all clients start at the beginning of the media file) can have a dramatic influence on the required server bandwidth.

The results in Figure 4 illustrate that as in the case of full-file access, the minimum required server bandwidth with interval access can be closely approached (within a small constant factor) with practical delivery protocols.

4.3. Random start/end

Consider now the case in which each requested interval has both a random start point, and a random end point. Suppose that the start position A of the requested interval is Uniform(0, T), and given $A = a$, the end position B is Uniform(a, T). Following the same methodology as that described above for the simpler cases, $E[Z|x]$ can be derived as

$$E[Z \mid x] = \frac{\lambda(T-x)}{T} \cdot \left\{ \begin{array}{l} \int_0^x z \ln\frac{T-x+z}{T-x} \cdot \exp(-\frac{T-x}{T}\lambda[(T-x+z)\ln\frac{T-x+z}{T-x} - z])dz + \\ \ln\frac{T}{T-x} \cdot \int_x^\infty z \cdot \exp(-\frac{T-x}{T}\lambda[(T-x+z)\ln\frac{T}{T-x} - x])dz \end{array} \right\}, \tag{22}$$

using which the minimum required server bandwidth can be numerically computed. It turns out that (21) still holds when $B_{min}^{random\ start}$ is replaced by $B_{min}^{random\ start\,/\,end}$ (see Appendix C), so the random end effect is negligible in terms of bandwidth savings when the request rate is high.

4.4. Marker-based access

Media files may have built-in markers that partition the content into sections for more convenient access (for example, as in the *eTeach* educational media server measured in [2]). Systems using markers may be analyzed using a similar approach as developed above. For example, consider the case of a media file with two markers with offsets 0 and a ($0 < a < T$), respectively, and each client request is for an interval of the media file beginning at either of the markers and continuing to the end of the file. Denote the probability of a requested interval starting at the first marker (i.e., a request for the full file) by p. In the following, such requests are called "type I requests", while those for the interval beginning at the second marker are called "type II requests".

A file segment at offset x between 0 and a can only be accessed by type I requests. Similar to the derivation of equation (11),

$$E[Z \mid 0 < x < a] = x + \frac{1}{p\lambda}. \tag{23}$$

For x between a and T, it is necessary to consider both the cases $z < x$ and z > x. Note that in the former case, $z < x$-a cannot occur, and only type II requests need be considered. The corresponding rates can be derived as:

$$\begin{aligned} \bar{\lambda}_{z<x}(x) &= \frac{z-(x-a)}{z}(1-p)\lambda \\ \tilde{\lambda}_{z<x}(x) &= (1-p)\lambda \quad , \end{aligned} \tag{24}$$

and

$$\begin{aligned} \bar{\lambda}_{z>x}(x) &= \frac{\lambda(z-x)+(1-p)\lambda(a-0)}{z} \\ \tilde{\lambda}_{z>x}(x) &= p\lambda \quad , \end{aligned} \tag{25}$$

yielding

$$E[Z \mid a < x < T] = \int_{x-a}^{x} z \exp(-\bar{\lambda}_{z<x} z)\tilde{\lambda}_{z<x} \mathrm{dz} + \int_{x}^{\infty} z \exp(-\bar{\lambda}_{z>x} z)\tilde{\lambda}_{z>x} \mathrm{dz}$$
$$= x - a + \frac{1}{(1-p)\lambda} - \exp[-(1-p)a\lambda]\frac{p}{(1-p)\lambda}. \qquad (26)$$

The above analysis can be extended in two ways. First, files that have more than two markers can be considered. Second, requests can be for an interval of the media file that begins at a marker (as before), but continues to an ending point specified by an arbitrary distribution function. The distribution function affects the calculations of the rates $\bar{\lambda}(x)$ and $\tilde{\lambda}(x)$, but not does not impact other aspects of the analysis, permitting considerable flexibility in the model.

Suppose *n* markers are at offsets 0 = $a_1 < a_2 < \ldots < a_n < T$, and let p_i be the probability that a request is for an interval starting at marker *i* (*i* = 1 ,…, *n*; p_i's sum to 1). Suppose that the ending point of a requested interval that begins at marker *i* has a cumulative distribution function $F_i(x)$, (*i* = 1 ,…, *n*). In addition, for a given infinitesimal file segment with offset *x*, define

$$\tilde{\lambda}_i = (1 - F_i(x))\, p_i \lambda \ , \qquad (27)$$

the arrival rate of requests for an interval beginning at marker *i* and including the segment with offset *x*. Given $x \in (a_i, a_{i+1})$ (*i* = 1 ,…, *n* and $a_{n+1} = T$),

$$\begin{aligned} E[Z \mid x] &= x - a_i + \frac{1}{\tilde{\lambda}_i} + \exp[-\tilde{\lambda}_i (a_i - a_{i-1})] \cdot \left(\frac{1}{\tilde{\lambda}_i + \tilde{\lambda}_{i-1}} - \frac{1}{\tilde{\lambda}_i}\right) \\ &+ \exp[-(\tilde{\lambda}_i + \tilde{\lambda}_{i-1})(a_{i-1} - a_{i-2}) - \tilde{\lambda}_i (a_i - a_{i-1})] \times \left(\frac{1}{\tilde{\lambda}_i + \tilde{\lambda}_{i-1} + \tilde{\lambda}_{i-2}} - \frac{1}{\tilde{\lambda}_i + \tilde{\lambda}_{i-1}}\right) \\ &+ \ldots + \exp[-(\tilde{\lambda}_i + \tilde{\lambda}_{i-1} + \ldots + \tilde{\lambda}_2)(a_2 - a_1) - (\tilde{\lambda}_i + \tilde{\lambda}_{i-1} + \ldots + \tilde{\lambda}_3)(a_3 - a_2) \\ &\quad - \ldots - \tilde{\lambda}_i (a_i - a_{i-1})] \times \left(\frac{1}{\tilde{\lambda}_i + \tilde{\lambda}_{i-1} + \ldots + \tilde{\lambda}_1} - \frac{1}{\tilde{\lambda}_i + \tilde{\lambda}_{i-1} + \ldots + \tilde{\lambda}_2}\right). \end{aligned} \qquad (28)$$

Figure 5 shows the lower bound on required server bandwidth for varying numbers of equally accessed markers, and for two models of stream durations. In one model, all clients play to the end of the file. In the other model, the stream endpoint is uniformly distributed between the starting position and the end of the file. Note that the required server bandwidth is very similar for both models of stream durations for any given number of markers. Also note that, although for a finite number of markers the required server bandwidth grows only logarithmically with the request rate, the constant factor increases as the number of markers increases, and thus as the number of markers increases, the required bandwidth approaches the required bandwidth for random start times given in equation (14) and also plotted in the figure.

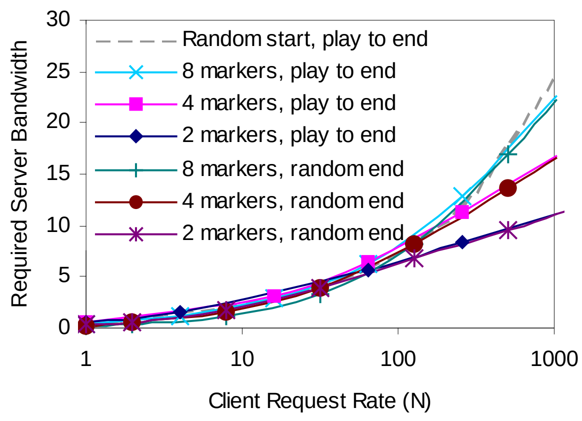


Figure 5: Minimum Required Server Bandwidth with Accesses at File Markers

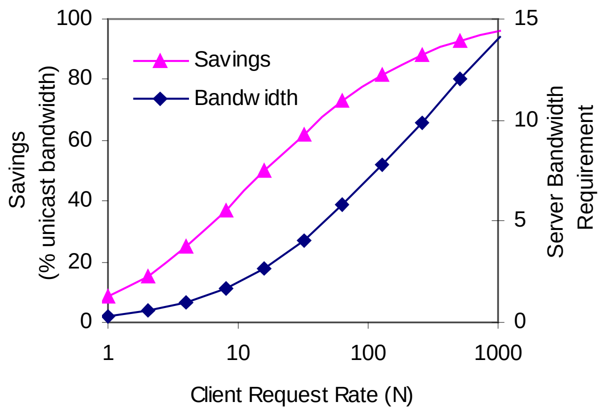


Figure 6: Required Bandwidth for 3 Equal Markers (stream endpoint uniform from start to T)

Figure 6 provides the lower bound on required server bandwidth as a function of client request rate, as computed from equation (28), for the case that the media file has markers at positions 0, T/3 and 2T/3, client requests start at one of the equally likely markers, and the stream endpoint is uniformly distributed between the starting position and the end of the media file. This model of interval requests is an idealized form of the typical profile of eTeach client requests illustrated in Figure 7(b). Figure 6 also shows the percent savings compared to unicast delivery for the required server bandwidth given in the figure. Note that for client request rates in the range of 15 to 75, bandwidth savings for scalable streaming compared with unicast streaming are in the range of 40% – 80% for this workload. Similar savings are computed in [2] from simulations of the HSM protocol and actual client requests in the eTeach logs, for files with request rates in this range and interval requests similar to those in Figure 7(b). Since the analytic calculations assume independent client requests, it appears likely that the bandwidth savings estimated in the simulations are not due to significant correlations in the client requests. Furthermore, it appears that the analytic estimates of server bandwidth, derived for Poisson arrivals, are not overly conservative for the eTeach client interarrival times which have somewhat higher variance

4.5. Worst case scaling of minimum required server bandwidth

We now address the question of whether access types exist for which immediate-service streaming media delivery can not be made scalable; i.e., for which we cannot achieve sub-linear scaling of required server bandwidth with request rate. The goal is to find the interval access pattern that has the largest asymptotic growth factor of minimum required server bandwidth with client request rate, under the assumption of Poisson and independent requests, for a given media file streaming rate and total duration. Although we restrict attention to independent, Poisson requests, this would not seem to be a significant restriction in practice. An arrival process burstier than Poisson would potentially improve (rather than degrade) the performance of scalable delivery protocols, as it would provide more opportunities for sharing transmissions to clients whose requests are closely spaced in time. Similarly, it would seem that correlations among requests with respect to which media file portions they require would most likely increase, rather than decrease, sharing opportunities.

A starting point is provided by further analysis of the case of two markers and access to the end of the file, as considered previously. Suppose that the second marker is precisely at the middle of the media file, at offset $a = T/2$, and consider the question of what value of p (i.e., division of the total arrival rate between the two markers) maximizes the lower bound on required server bandwidth for large N. From equations (23) and (26),

$$\begin{aligned} B_{min}^{two\ \ markers} &= \int_0^{T/2} \frac{\mathrm{d}x}{E[Z \mid 0 < x < \frac{T}{2}]} + \int_{T/2}^{T} \frac{\mathrm{d}x}{E[Z \mid \frac{T}{2} < x < T]} \\ &= \ln(\frac{N_1}{2} + 1) + \ln(\frac{N_2/2}{1 + \exp(-N_2/2)(N_2/N - 1)} + 1), \end{aligned} \tag{29}$$

where $N_1 = p\lambda T$ and $N_2 = (1\text{-}p)\lambda T$ are the average numbers of requests for the intervals beginning at the respective markers within a time period of length T ($N_1+N_2=N$).[3] For any $\delta > 0$, it holds that

$$\lim_{\substack{N \to \infty \\ N_2/N > \delta}} \frac{B_{min}^{two\ \ markers}}{\ln(N_1/2) + \ln(N_2/2)} = 1. \tag{30}$$

On the other hand, by the arithmetic-geometric mean inequality,

$$\ln\frac{N_1}{2} + \ln\frac{N_2}{2} = \ln\frac{N_1 N_2}{4} \le \ln\frac{(N_1+N_2)^2}{16} = \ln\frac{N^2}{16}. \tag{31}$$

and the equality holds if and only if $N_1 = N_2$, or $p = 1/2$. Since $\ln(N^2/16) > \ln(N+1)$ when N is sufficiently large, the degenerate case of $p \to 1$ can be precluded from further consideration. Since δ is arbitrary, it can be inferred from (30) and (31) that splitting requests equally between the two markers results in the worst-case minimum required server bandwidth asymptotically with respect to N.

The above result is readily extended to the case where there are n equally spaced markers in a media file. The explicit form of $B_{min}^{n\ \ markers}$ can be derived using (28). Note that the exponents will vanish with large N_i's. The analogous result to (30) for the case of n markers is

$$\lim_{\substack{N \to \infty \\ N_i/N > \delta \\ i=2,\ldots,n}} \frac{B_{min}^{n\ \ markers}}{\ln(N_1/n) + \ln(N_2/n) + \cdots + \ln(N_n/n)} = 1. \tag{32}$$

Similarly, (31) should now be rewritten as

$$\ln\frac{N_1}{n} + \ln\frac{N_2}{n} + \cdots + \ln\frac{N_n}{n} = \ln\frac{N_1 N_2 \cdots N_n}{n^n} \le \ln\frac{(N_1 + N_2 + \cdots + N_n)^n}{n^{2n}} = n\ln\frac{N}{n^2}, \tag{33}$$

and the equality holds if and only if all the N_i's are equal; i.e., access is uniform over all markers. Thus, again, splitting requests equally between the markers results in the worst-case minimum required server bandwidth asymptotically with respect to N.

[3] Note that the limit of the above expression as $p \to 1$ is $\ln(N+1)$, consistent with equation (1).

Any access type can be approximated arbitrarily closely by requests for intervals starting at equally-spaced (but, in general, not equally-popular) markers, by choosing the number of markers to be sufficiently large. Further, the required server bandwidth can only be increased if a requested interval is lengthened, and thus the worst case with respect to the minimum required server bandwidth is achieved when requests are for intervals that continue to the end of the file. Finally, the worst case minimum required server bandwidth must increase for larger numbers of markers (asymptotically with respect to N), as (for example) n equally-spaced markers are a subset of $2n$ equally-spaced markers, and in the latter case the worst case minimum required server bandwidth is achieved when all $2n$ markers have equal request rate. These points imply that the minimum required server bandwidth is maximized, asymptotically with respect to N, for intervals that have a random (uniformly distributed) starting point, and continue to the end of the media file. From equation (21), we then have that

$$B_{min}^{worst\ case} \sim \sqrt{\frac{2}{\pi} N}\,. \tag{34}$$

An important point with respect to scalability in practice, however, is that for any fixed number of markers, the worst-case minimum required server bandwidth is still logarithmic in N. In other words, for any positive integer $n > 1$, there exists a constant C_n (the subscript n implies that the constant depends on n) such that

$$\lim_{N \to \infty} \frac{B_{min}^{n\ markers}}{C_n \ln N} = 1\,. \tag{35}$$

This result is consistent with equation (34), since, by equations (32) and (33), equation (35) can only be satisfied by setting $C_n = n$, which implies that

$$\lim_{n \to \infty} C_n = \infty\,. \tag{36}$$

This asserts that no logarithmic growth can be sustained as n increases, although the growth for every single n is logarithmic.

Another point to note is that the above analysis also yields insight into the worst-case ratio of the minimum required server bandwidth with scalable streaming, to the bandwidth with unicast. For any *fixed* average request size, the bandwidth with scalable streaming (assuming Poisson, independent requests) asymptotically grows at least a factor of $\sqrt{N}$ slower, as N increases, than does the required server bandwidth with unicast delivery.

5. Scaling system size

Previous analyses of scalable streaming have determined how the required server bandwidth scales as the request rate for a media file is increased. However, media delivery systems may scale up by serving more files, which then leads to higher total request rates. A natural question to consider is where the bandwidth cost will lie in large-scale systems; i.e., will this cost be dominated by the bandwidth needed for relatively cold files for which multicast delivery is only minimally effective. This might be the case, for example, if the distribution of file popularities is heavy-tailed.

To gain some insight into this question, consider a media delivery system that scales in both the number of served files, and the total request rate, and that provides immediate service to each request.

Denote the request rate to the least popular file by λ_{min}. An assumption is needed with respect to how λ_{min} changes as the media delivery system is scaled up. Of interest here are systems in which each file has (asymptotically) non-zero request rate; i.e., λ_{min} is bounded away from zero. Thus, in analyzing the system scaling behaviour, we can either assume that λ_{min} does not scale (i.e., is effectively constant), or that it scales up, as system size is scaled. We make here the conservative (in the sense of implying higher required server bandwidth) assumption that λ_{min} is (an arbitrarily small) constant. In this case, assuming a Zipf distribution of object popularities, in a system with K total files, total request rate λ, and request rate for the least popular file of λ_{min},

$$\lambda \frac{1}{K\sum_{i=1}^{K} 1/i} = \lambda_{min} \ , \tag{37}$$

implying that the (simultaneous) scaling of λ and K is such that

$$\lambda \approx \lambda_{min} K \ln K \ . \tag{38}$$

Assume full-file playback, files of equal playback duration T, and Poisson request arrivals. From the lower bound on required server bandwidth given in equation (1), and assuming a Zipf distribution of file popularities, a tight lower bound on the required server bandwidth for all files is given by

$$B_{min}^{all} = \sum_{i=1}^{K} \ln\left(\frac{KT\lambda_{min}}{i} + 1\right) \approx \ln\left(\frac{(KT\lambda_{min})^K}{K!}\right) \approx \ln\left(\frac{(eT\lambda_{min})^K}{\sqrt{2\pi K}}\right). \tag{39}$$

This last expression (derived from Stirling's approximation for factorial) implies that a lower bound on the required server bandwidth with scalable streaming is $O(K)$. Since the required server bandwidth with hierarchical stream merging, for example, is of the same order as the lower bound from Section 2, $O(K)$ required server bandwidth is achievable. With unicast delivery, in contrast, the required server bandwidth is $O(\lambda)$, or, from expression (38), $O(K \ln K)$. This suggests that scalable streaming can yield significant benefits even in this conservative scenario in which λ_{min} is fixed, and thus (owing to the heavy-tailed Zipf distribution) the number of files being served is scaling relatively rapidly with the system size.

Suppose now that accesses to each file are for an arbitrary interval rather than the full-file. From the results in Section 4, in the worst case the minimum required server bandwidth is

$$O\left(\sum_{i=1}^{K} \sqrt{\frac{KT\lambda_{min}}{i}}\right). \tag{40}$$

Interestingly, this expression is also $O(K)$, and since it appears that the required server bandwidth with hierarchical stream merging is of the same order as that of the lower bound in this context as well, our conclusion in a context of interval access is similar to that for full-file access – it appears that scalable streaming may be able to yield significant benefits even in this conservative system scaling scenario.

6. Conclusions

This paper has studied the robustness of the performance benefits of scalable on-demand streaming by considering three important contexts in which the benefit might be small. In one such context, clients can tolerate appreciable delays on the order of the playback duration, and therefore scalable on-demand streaming might not yield sufficiently improved performance in comparison to simpler scalable download techniques. A second context concerns systems in which client requests are for media file intervals rather than full files, and thus in which there may be less opportunity for sharing transmissions of media file data among many clients. The third context concerns large media servers with many files, in which a significant fraction of the server bandwidth might be devoted to serving relatively cold objects for which scalable streaming is not effective.

Our results suggest that scalable streaming can yield substantial benefits in each of these contexts, and thus that this technology may be effective in a wide range of environments where media content is sufficiently popular (i.e., sustains sufficiently high request rates). Our results also quantify the benefit in each of the contexts studied, including the determination of the worst-case scaling, of the minimum required server bandwidth with request rate, for immediate service with independent interval requests rather than full-file accesses.

Future work includes designing content distribution networks for popular media content under a variety of client access assumptions, analyzing further existing streaming server workloads, investigating new models of client interactive access for future streaming applications, and devising new protocols for scalable streaming over the Internet.

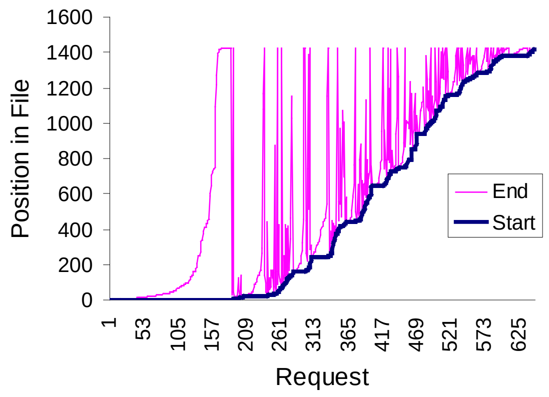
Acknowledgments

We thank Jussara Almeida for the profiles of client interval requests from the eTeach server that are presented in Appendix A of the paper, and Prof. Yongji Tan at Fudan University for his advice on the mathematical techniques that we used to derive the results in Appendices B and C.

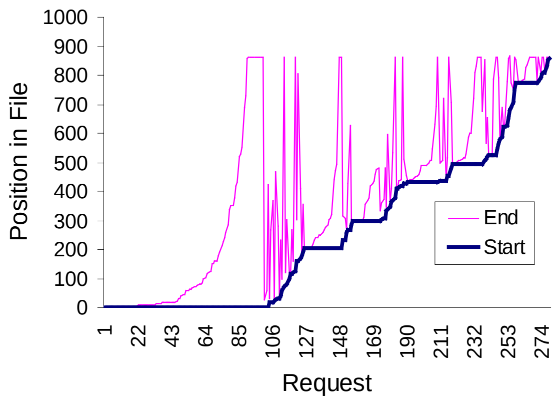
Appendix A. Measured media server interval requests

A recent study that characterizes a one-month workload for an educational media server called *eTeach* at the University of Wisconsin [2], showed that while the server received over one thousand requests on many weekdays, the average stream duration is just two minutes. Each ten-second portion of the most frequently requested lectures was viewed approximately equally often, but students viewed only a minute or two at a time, possibly pausing to take notes.

Two different profiles of the intervals that were requested from the *eTeach* server are shown in Figure 7. Each of these profiles is typical for many different files in the trace. In each plot, the requests that occurred for the file during the day are sorted by the start position of the interval, and then by the end position. The lower smoother curve plots the start positions while the upper curve plots the end positions of the requested intervals. In Figure 7(a) the interval start position is either the beginning of the file (for about 200 of the requests) or is approximately uniformly distributed over the length of the video, and the interval durations are highly variable. In Figure 7(b) many of the interval start times occur at a few markers in the video (i.e., positions in the video that clients access by clicking on the outline for the lecture). In a third typical interval access pattern observed (but not



(a) Interval start times are either at position 0 or are approximately uniform in (0,T) (most popular video on Sep 27)



(b) Clients frequently start at markers in video (most popular file on Oct 10)

Figure 7: Measurements of Video Interval Requests in Eteach

shown to conserve space), nearly all requests start at the beginning of the video, a large fraction of the requests are for the full file, and the remaining requests end at positions approximately uniformly distributed between zero and the file duration, *T*. These access patterns motivate some of the interval access distributions that are considered in Section 4.

Appendix B

This appendix provides a proof of equation (21). For clarity we assume that *T* is a constant, and therefore that $N \to \infty$ is equivalent to $\lambda \to \infty$. By (20), the left hand side of (21) can be written as

$$\lim_{\lambda\to\infty}\int_0^1 \sqrt{\frac{\pi}{2}} \cdot \frac{\mathrm{dy}}{\sqrt{2\pi}\Phi(\sqrt{N}y) + \frac{1}{y\sqrt{\lambda}}\exp(-\frac{N}{2}y^2)} . \tag{B.1}$$

Since the integrand is uniformly continuous about $N \in [\alpha, \infty)$ for any given α>0, the order of limit and integration can be exchanged. Given the fact that $\lim_{t\to\infty} \Phi(t) = 1/2$, (B.1) can be rewritten as

$$\int_0^1 \lim_{N\to\infty} \sqrt{\frac{\pi}{2}} \cdot \frac{dy}{\sqrt{2\pi}\Phi(\sqrt{N}y) + \frac{1}{y\sqrt{\lambda}}\exp(-\frac{N}{2}y^2)} = \int_0^1 \sqrt{\frac{\pi}{2}} \cdot \frac{dy}{\sqrt{2\pi}\cdot\frac{1}{2}} = 1, \tag{B.2}$$

which completes the proof.

Appendix C

As in Appendix B, we assume that *T* is a constant, and therefore $N \to \infty$ is equivalent to $\lambda \to \infty$. The goal is to show that

$$\lim_{\lambda\to\infty} \frac{B_{\min}^{randomstart/end}}{\sqrt{\frac{2}{\pi}N}} = \lim_{\lambda\to\infty} \int_0^T \sqrt{\frac{\pi}{2\lambda T}} \cdot \frac{dx}{E[Z \mid x]} = 1, \tag{C.1}$$

where E[Z|x] can be substituted by (22). Note that equation (22) can be further simplified. In particular, *E*[*Z*|*x*] = *I* + *II*, where

$$I = \frac{\lambda(T-x)}{T} \cdot \int_0^x z \ln\frac{T-x+z}{T-x} \cdot \exp(-\frac{T-x}{T}\lambda[(T-x+z)\ln\frac{T-x+z}{T-x} - z])dz, \tag{C.2}$$

and

$$II = \frac{\lambda(T-x)}{T}\ln\frac{T}{T-x} \cdot \int_x^\infty z \cdot \exp(-\frac{T-x}{T}\lambda[(T-x+z)\ln\frac{T}{T-x} - x])dz. \tag{C.3}$$

In particular, II can be explicitly solved:

$$II = \exp(\frac{T-x}{T}\lambda x) \cdot \exp[-(T-x)\lambda \ln\frac{T}{T-x}] \cdot [x + \frac{T}{(T-x)\lambda} \cdot \frac{1}{\ln\frac{T}{T-x}}]$$

$$= \exp[(T-x)\lambda(\frac{x}{T} - \ln\frac{T}{T-x})] \cdot [x + \frac{T}{(T-x)\lambda} \cdot \frac{1}{\ln\frac{T}{T-x}}]. \tag{C.4}$$

Thus the left hand side of (C.1) equals

$$\lim_{\lambda\to\infty} \int_0^T \sqrt{\frac{\pi}{2\lambda T}} \frac{dx}{E[Z \mid x]} = \lim_{\lambda\to\infty} \int_0^T (\frac{1}{\sqrt{\frac{2\lambda T}{\pi}} \times I + \sqrt{\frac{2\lambda T}{\pi}} \times II})dx. \tag{C.5}$$

For any given *x* (0<*x*<*T*), $\lim_{\lambda\to\infty}\sqrt{\frac{2}{\pi}\lambda T} \times II = 0$ because $\frac{x}{T} - \ln\frac{T}{T-x} < 0$ (This can be shown by letting $f(x) = \frac{x}{T} - \ln\frac{T}{T-x}$ and observing that *f*(0)=0, *f'*(*x*)<0 for any *x* in the range) . Therefore more attention should be drawn to the first term in the denominator of the integrand in (C.5).

It is necessary to evaluate

$$\lim_{\lambda\to\infty} \sqrt{\frac{2}{\pi}\lambda T} \cdot \frac{\lambda(T-x)}{T} \cdot \int_0^x z \ln\frac{T-x+z}{T-x} \cdot \exp(-\frac{T-x}{T}\lambda[(T-x+z)\ln\frac{T-x+z}{T-x} - z])dz. \tag{C.6}$$

Let $w = \sqrt{\frac{\lambda}{T}}z$ and rewrite (C.6) as

$$\lim_{\lambda\to\infty}\sqrt{\frac{2}{\pi}}\cdot\lambda(T-x)\cdot\int_0^{\sqrt{\frac{\lambda}{T}}x} w\ln(1+\sqrt{\frac{T}{\lambda}}\frac{w}{(T-x)})\cdot\exp\{-\frac{T-x}{T}\lambda[(T-x+\sqrt{\frac{T}{\lambda}}w)\ln(1+\sqrt{\frac{T}{\lambda}}\frac{w}{(T-x)})-\sqrt{\frac{T}{\lambda}}w]\}\sqrt{\frac{T}{\lambda}}dw. \quad \text{(C.7)}$$

The integrand in (C.7) is uniformly continuous about $\lambda\in[\lambda_0,\infty)$ for any given $\lambda_0>0$ (also note that the exponent in the integrand above is negative such that the integrand itself can be arbitrarily close to 0 as long as either w or λ is sufficiently large). Hence, the order of limit and integration can be exchanged. Also by Taylor's expansion, for any $w>0$,

$$\begin{aligned}
&\lim_{\lambda\to\infty}\sqrt{\frac{2\lambda T}{\pi}}(T-x)w\ln(1+\sqrt{\frac{T}{\lambda}}\frac{w}{(T-x)})\cdot\exp\{-\frac{T-x}{T}\lambda[(T-x+\sqrt{\frac{T}{\lambda}}w)\ln(1+\sqrt{\frac{T}{\lambda}}\frac{w}{(T-x)})-\sqrt{\frac{T}{\lambda}}w]\}\\
&\quad=\lim_{\lambda\to\infty}\sqrt{\frac{2\lambda T}{\pi}}(T-x)\sqrt{\frac{T}{\lambda}}\frac{w^2}{(T-x)}\cdot\exp\{-\frac{T-x}{T}\lambda\cdot\frac{Tw^2}{2\lambda(T-x)}\} \qquad \text{(C.8)}\\
&\quad=\lim_{\lambda\to\infty}\sqrt{\frac{2}{\pi}}Tw^2\exp(-\frac{w^2}{2})=\sqrt{\frac{2}{\pi}}Tw^2\exp(-\frac{w^2}{2}).
\end{aligned}$$

Thus (C.7) equals

$$\int_0^\infty\sqrt{\frac{2}{\pi}}Tw^2\cdot\exp(-\frac{w^2}{2})dw=\int_0^\infty\sqrt{\frac{2}{\pi}}T\cdot\exp(-\frac{w^2}{2})dw=2T\int_0^\infty\frac{1}{\sqrt{2\pi}}\exp(-\frac{w^2}{2})dw=2T\cdot\frac{1}{2}=T. \quad \text{(C.9)}$$

The result above also confirms that the integrand in (C.5) is uniformly continuous about $\lambda\in[\lambda_0,\infty)$, so that the order of limit and integration in (C.5) are exchangeable. When the limit operation is taken first, (C.5) equals

$$\int_0^T\frac{dx}{T+0}=1, \qquad \text{(C.10)}$$

which completes the proof.

References

[1] S. Acharya, M. J. Franklin, and S. Zdonik, "Dissemination-based Data Delivery Using Broadcast Disks", *IEEE Personal Communications 2*, 6 (Dec. 1995), pp. 50-60.

[2] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon, "Analysis of Educational Media Server Workloads", *Proc. NOSSDAV 2001*, Port Jefferson, NY, June 2001.

[3] M. Ammar and J. Wong, "The Design of Telextext Broadcast Cycles", *Performance Evaluation* 5, 4 (Nov. 1985), pp. 235-242.

[4] A. Bar-Noy, G. Goshi, R. E. Ladner, and K. Tam, "Comparison of Stream Merging Algorithms for Media-on-Demand", *Proc. MMCN 2002*, San Jose, CA, Jan. 2002.

[5] Y. Birk and R. Mondri, "Tailored Transmissions for Efficient Near-Video-On-Demand Service", *Proc. IEEE ICMCS'99*, Florence, Italy, June 1999.

[6] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", *Proc. ACM SIGCOMM'98*, Vancouver, Canada, Sept. 1998.

[7] M. Chesire, A. Wolman, G. Voelker, and H. M. Levy, "Measurement and Analysis of a Streaming Media Workload", *Proc. USITS 2001*, San Francisco, CA, Mar. 2001.

[8] Y. Cai, K. A. Hua, and K. Vu, “Optimizing Patching Performance”, *Proc. MMCN’99,* San Jose, CA, Jan. 1999.

[9] S. W. Carter and D. D. E. Long, “Improving Video-on-Demand Server Efficiency Through Stream Tapping”, *Proc. ICCCN’97*, Las Vegas, NV, Sept. 1997.

[10] E. G. Coffman, Jr., P. Jelenkovic, and P. Momcilovic, “Provably Efficient Stream Merging”, *Proc. 6th Int’l. Workshop on Web Caching and Content Distribution*, Boston, MA, June 2001.

[11] D. L. Eager, M. K. Vernon, and J. Zahorjan, “Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand”, *Proc. MMCN 2000,* San Jose, CA, Jan. 2000.

[12] D. L. Eager, M. K. Vernon, and J. Zahorjan, “Minimizing Bandwidth Requirements for On-Demand Data Delivery”, *IEEE Trans. on Knowledge and Data Engineering 13,* 5 (Sept./Oct. 2001), Special Section of invited papers from MIS’99, pp. 742-757.

[13] D. L. Eager, M. K. Vernon and J. Zahorjan, “Optimal and Efficient Merging Schedules for Video-on-Demand Servers”, *Proc. ACM MULTIMEDIA’99*, Orlando, FL, Nov. 1999.

[14] L. Gao and D. Towsley, “Supplying Instantaneous Video-on-Demand Services Using Controlled Multicast”, *Proc. IEEE ICMCS’99,* Florence, Italy, June 1999.

[15] A. Hu, “Video-on-Demand Broadcasting Protocols: A Comprehensive Study”, *Proc. IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001.

[16] K. A. Hua and S. Sheu, “Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems”, *Proc. ACM SIGCOMM’97,* Cannes, France, Sept. 1997.

[17] K. A. Hua, Y. Cai, and S. Sheu, “Patching: A Multicast Technique for True Video-On-Demand Services”, *Proc. ACM MULTIMEDIA’98,* Bristol, U.K., Sept. 1998.

[18] S. Jin and A. Bestavros, “Scalability of Multicast Delivery for Non-sequential Streaming Access”, *Proc. ACM SIGMETRICS 2002*, Marina Del Rey, CA, June 2002.

[19] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel, "Scalable On-Demand Media Streaming with Packet Loss Recovery", *Proc. ACM SIGCOMM 2001*, San Diego, CA, Aug. 2001.

[20] L. Rizzo and L. Vicisano, “A Reliable Multicast Data Distribution Protocol Based on Software FEC Techniques”, *Proc. HPCS’97*, Chalkidiki, Greece, June 1997.

[21] S. Sen, L. Gao, and D. Towsley, "Frame-Based Periodic Broadcast and Fundamental Resource Tradeoffs", Technical Report 99-78, Computer Science Dept., U. Mass. – Amherst, 1999.

[22] H. Tan, D. L. Eager, M. K. Vernon, and H. Guo, “Quality of Service Evaluations of Multicast Streaming Protocols”, *Proc. ACM SIGMETRICS 2002,* Marina del Rey, CA, June 2002.

[23] S. Viswanathan and T. Imielinski, “Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting”, *Multimedia Systems 4*, 4 (Aug. 1996), pp. 197-208.

[24] Y. Zhao, D. L. Eager, and M. K. Vernon, "Network Bandwidth Requirements for Scalable On-Demand Streaming”, *Proc. IEEE INFOCOM 2002,* New York, NY, June 2002.