Quality of Service Evaluations of Multicast Streaming Protocols*

Haonan Tan◆ Derek L. Eager¤ Mary K. Vernon◆ Hongfei Guo◆

◆Computer Sciences Department
University of Wisconsin-Madison, USA
{haonan, vernon, guo}@cs.wisc.edu

¤Department of Computer Science
University of Saskatchewan, Canada
eager@cs.usask.ca

ABSTRACT

Recently proposed scalable on-demand streaming protocols have previously been evaluated using a system cost measure termed the "required server bandwidth". For the scalable protocols that provide immediate service to each client when the server is not overloaded, this paper develops simple analytic models to evaluate two client-oriented quality of service metrics, namely (1) the mean client waiting time in systems where clients are willing to wait if a (well-provisioned) server is temporarily overloaded, and (2) the fraction of clients who balk (i.e., leave without receiving their requested media content) in systems where the clients will tolerate no or only very low service delays during a temporary overload. The models include novel approximate MVA techniques that appear to extend the range of applicability of customized AMVA to include questions focussed on state probabilities rather than on mean values, and to systems in which the operating points of interest do not include substantial client queues. For example, the new AMVA models accurately estimate the server bandwidth needed to achieve a balking rate as low as one in ten thousand. The analytic models can easily be applied to determine the server bandwidth needed for a given number of media files, anticipated total client request rate and file access frequencies, and target balking rate or mean wait. Results show that (a) scalable media servers that are configured with the "required server bandwidth" defined in previous work have low mean wait but may have unacceptably high client balking rates (i.e., greater than one in twenty), (b) for high to moderate client load, only a 10 – 50% increase in the previously defined required server bandwidth is needed to achieve a very low balking rate (e.g., one in ten thousand), and (c) media server performance (either mean wait or balking rate) degrades rapidly if the actual client load is more than 10% greater than the anticipated load.

1. INTRODUCTION

A problem of considerable current interest is that of providing scalable Internet services to potentially vast numbers of clients. Particularly challenging are scalable delivery protocols for applications that require on-demand streaming of multimedia, such as news services, distance education, or entertainment-on-demand. Recently proposed protocols [3, 5-7, 9-17, 20, 22, 23] use broadcast or multicast to reduce the required server and network bandwidth from linear in the request rate, to sublinear. They achieve this bandwidth reduction by dynamically aggregating clients that make requests closely spaced in time, so that eventually these clients share the same stream(s).

To appear in *Proc. ACM SIGMETRICS 2002 Int'l. Conf. On Measurement and Modeling of Computer Systems*, Marina del Rey, CA, June 2002.

Three of the recent multicast streaming protocols, namely patching [5, 6, 14, 16, 22], dynamic skyscraper [9, 11], and hierarchical stream merging (HSM) [3, 7, 10-12] provide scalable on-demand streaming without requiring clients to wait for some period of time as in a periodic broadcast system [13, 15, 17, 20, 23] or a batching system. These three protocols and a variant of HSM called "bandwidth skimming" [10] have been compared using a server cost or provisioning measure termed the *required server bandwidth* [11, 12]. This measure is the average concurrent server bandwidth used if the server has infinite bandwidth and each client request is satisfied immediately. The previous results show that the required server bandwidth for HSM systems, for full file requests and very general assumptions about the client arrival process, grows only with the logarithm of the client request rate, whereas the required server bandwidth for patching systems with Poisson arrivals and full file requests grows with the square root of the client request rate [11, 14]. In practical terms, if clients request the most popular media content at a rate greater than 50 requests per time that it takes to play the content, the HSM system has lower required server bandwidth than patching. The dynamic skyscraper system has required server bandwidth that grows with the logarithm of the client request rate, but with a larger constant factor than for HSM. Since HSM also supports interactive client requests more efficiently than the dynamic skyscraper protocol, we consider only the HSM and patching protocols in this paper.

A key open question addressed in this paper is how well the patching and HSM systems perform when the server has fixed (well provisioned) bandwidth and two client-oriented, "quality of service" metrics are considered, namely (1) the mean client wait if clients wait when the server is temporarily overloaded, and (2) the fraction of clients who leave without receiving their requested media content if clients will tolerate no or only low service delays during the temporary overload. The latter quantity is called the "balking rate" in the remainder of this paper. Specifically, for each protocol, we investigate the following questions:

1. What is the mean waiting time, or the client balking rate, when the server is configured with the required server bandwidth defined above for a given set of popular media objects and a given client load?

* This work was partially supported by the U. S. National Science Foundation under grants CCR-9975044, ANI-0117810, and EIA-0127857, and by the Natural Sciences and Engineering Research Council of Canada under Grant OGP-0000264.

2. How much additional or less server bandwidth is needed to achieve a given target client balking rate, such as one in ten thousand, or a given target mean waiting time, such as $0.001T$, where T is the requested media playing time?
3. How sensitive is the mean wait, or balking rate, to client arrival rate higher than the anticipated rate that the server is configured for?

For patching systems these questions are particularly relevant for modest client loads, in which case the previously defined required server bandwidths for patching and HSM are similar.

A key goal is to evaluate the QoS measures over a wide range of system configurations and client workloads. Simulation is time-consuming, and error-prone (e.g., due to lack of validation of the streaming protocol implementation, difficulty in choosing simulation run lengths for estimating low balking rates, etc.). Thus, another question addressed in this paper is whether simple, efficient analytic methods can be developed to accurately estimate the QoS metrics of interest.

Previous work [8] has proposed analytic methods for estimating mean wait and balking rate for relatively simple batching systems (in which full media file transmissions start at fixed times) and for FCFS unicast streaming systems. Developing accurate analytic methods for the patching and HSM systems considered here is substantially more complex for at least three reasons. First, the multicast stream duration (or service time) is correlated with the number of active streams for the object, and the precise form of the correlation is not easy to characterize. Second, the client balking rate cannot be computed using a known waiting time distribution or known stream start times. Third, when clients wait for service, the clients waiting for the same media file batch together for a stream with a state-dependent start time.

Previous techniques that accurately model state-dependent behavior such as general state-dependent service times, client balking, or clients selectively batching while waiting for service, are primarily based on state-space analysis (e.g., Markov Chain analysis), which has high computational cost for large and complex systems. Alternatively, Approximate Mean Value Analysis (AMVA) techniques have the advantages of very low computational cost and intuitive equations that readily permit heuristic extensions for modeling complex system features. However, it is not clear that AMVA can be applied to systems where clients batch while waiting for service or to the state-dependent stream durations that occur in scalable streaming systems. Furthermore, AMVA seems to be inherently focused on predicting the impact of queueing on mean performance metrics such as mean waiting times or mean queue lengths. It is not clear *a priori*, that AMVA techniques can be applied to questions such as how to provision a system so that the probability that a client has to wait is close to zero. Such questions may become more important (for example) as Internet services mature and become more utility-like. By developing customized AMVA techniques for estimating mean wait or client balking rate in on-demand streaming systems, we hope to obtain (a) insight into scalable media server provisioning questions, and (b) specific techniques and approaches that may be applicable in other contexts.

The main contributions of the paper are:

- Efficient, highly accurate *customized* AMVA models for estimating mean client waiting time or client balking rate in patching and HSM systems.
- A server configured with the previously defined "required server bandwidth" for a given protocol and anticipated moderate to high client arrival rate will have mean client wait under $0.015T$ for HSM or under $0.025T$ for patching, where T is the time to play the media file.
- A patching or HSM system configured with the previously defined "required server bandwidth" will have unacceptably high client balking rate (i.e., greater than one in twenty) if clients are unwilling to wait for service. However the results also show that very low balking rate (e.g., 1/10,000) can be achieved with a relatively modest increase in server bandwidth (i.e., on the order of a 10% – 50% increase for high to moderate client load).
- Scalable media server performance (either mean wait or balking rate) degrades rapidly if the actual client load is more than 10% greater than the anticipated load.

The models for client balking rate include customized AMVA techniques that are capable of accurately estimating the server bandwidth needed to achieve a wide range of target balking rates (from 10^{-1} to below 10^{-4}, for example), suggesting that customized AMVA techniques may be applicable to a significantly broader range of metrics and system design questions than has previously been recognized. The models for mean client waiting time, also based on AMVA, are also interesting in that simple models are found to be quite accurate, even though scalable streaming systems have a number of complex features (such as stream length being strongly correlated with the number of active streams for a file, and batching of requests for the same file while waiting in a queue) that might suggest a need for more complex models.

The remainder of the paper is organized as follows. Section 2 provides background on patching, hierarchical stream merging, the "bandwidth skimming" variant of HSM, and the previous analysis of required server bandwidth for these protocols. Sections 3 and 4 develop models for estimating client balking rate and mean client waiting time, respectively. Section 5 validates the models and applies them to evaluate patching and HSM system configurations with respect to the client QoS metrics. Conclusions are presented in Section 6.

2. BACKGROUND

The notation used in this section and in Section 3 is summarized in Table 1. In this section we review the patching protocol (Section 2.1), the HSM and bandwidth skimming protocols (Section 2.2), and the system assumptions that are made in developing the new models in this paper (Section 2.3).

2.1 Patching

The basic operation of patching [5, 6, 14, 16, 22] is as follows.[1] In response to a given client request, the server delivers the requested media in a single multicast stream. A client that submits a new request for the same file soon after this stream has started listens to the multicast, buffering the data received. Each such client is also provided a new unicast stream (i.e., a "patch" stream) that delivers the data that was delivered in the multicast stream prior to

[1] Our protocol descriptions and analyses assume that clients have sufficient local storage to buffer all data received in advance of its playback time. The protocols can easily accommodate client buffer constraints, as described in the cited earlier work.

the new client's request. The patch stream terminates when it reaches the point where the client joined the full-file multicast. Both the multicast stream and the patch stream deliver data at the file play rate so that each client can play the file in real time. Thus, the achievable aggregate transmission rate to a client must be at least twice the file play rate.

To keep the patch streams short, when the fraction of the file delivered in the most recent multicast exceeds a given threshold, the next client request triggers a new multicast rather than a patch stream. In "optimized" patching [6, 14], the threshold for each file is chosen to minimize the *required server bandwidth* (as defined in Section 1), assuming known file request rate, Poisson arrivals, and that each client requests the full file. We analyze optimized patching in this performance study, but note as in previous work that optimal choice of the threshold parameter is difficult to achieve in practice. Letting N_i denote the average number of requests for file i that arrive during the file play time T_i, the optimal threshold is $(\sqrt{2N_i+1}-1)/N_i$, and the required server bandwidth for delivery of file i, in units of the media play rate, can be derived for Poisson request arrivals as [14]:

$$B^*_{i,patching}(N_i) = \sqrt{2N_i+1} - 1. \quad (1)$$

2.2 HSM and Bandwidth Skimming

Hierarchical stream merging protocols [3, 7, 10-12] initiate a new multicast transmission stream for each new client request. In the simplest case, each client also listens to the closest earlier stream for the file that is still active [12], so that its own stream can terminate after transmitting the data that was missed in the earlier stream, as illustrated in Figure 1. In the figure, clients A through D request the media file at times T1, T2, T3, and T4, respectively. At time T4, client D listens to the stream that starts at T4 as well as the stream that was initiated for client C at time T3. At time T5, the stream for client D can be terminated, and clients C and D are "merged". When clients merge, they listen to the closest earlier stream that is still active, and so on. A number of more complex HSM policies have very similar performance [3, 7, 12].

Bandwidth skimming is a variant of hierarchical stream merging that uses only a small "skim" of the achievable transmission rate to a client to support the merging. This is useful in cases where the quality of the media content is such that a single stream at the play rate consumes more than half of the achievable transmission bandwidth to the client. Bandwidth skimming policies [10] use the hierarchical stream merging strategy together with a mechanism for effecting each merge that does not require clients to listen to two play rate streams concurrently. For example, in the "Partition" policy, streams are divided into substreams, and a client A merges with another client B by listening to successively increasing numbers of B's substreams, while listening to successively decreasing numbers of its own substreams.

Letting b denote the required aggregate transmission rate to a client, in units of the media play rate, the required server bandwidth for a given file under HSM (b = 2) or bandwidth skimming (b < 2) and Poisson request arrivals can be approximated by the following formula [11]:

$$B^*_{i,HSM/bandwidth\ skimming}(N_i) \approx \eta_b \ln\left(\frac{N_i}{\eta_b}+1\right), \quad (2)$$

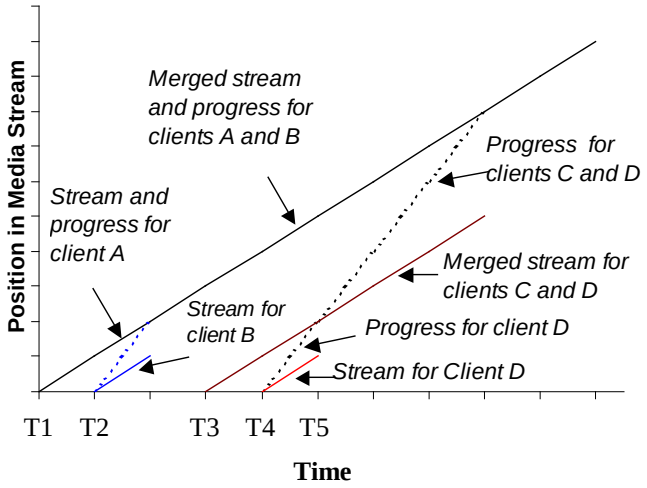


Figure 1: HSM Example

where η_b is the positive real constant that satisfies the following equation:

$$\eta_b\left(1-\left(\frac{\eta_b}{\eta_b+1}\right)^b\right)=1.$$

For example, for HSM with b=2, the required server bandwidth can be reasonably approximated by [11]:

$$B^*_{i,HSM}(N_i) \approx 1.62\ln(N_i/1.62+1). \quad (3)$$

2.3 QoS Motivation and System Assumptions

Figure 2 plots the required server bandwidth for a media file as a function of client request rate for the file, N_i, as computed from equations (1) – (3) above. Previous work [12] has provided the rationale as well as results from simulations that show that an HSM server containing a collection of media files with given client request rates has average client waiting time "close to zero" if configured to support a maximum number of concurrent streams equal to the sum of the required bandwidth for each file given in equation (3).

This paper develops, validates, and applies models to determine more precisely, for patching and bandwidth skimming servers as well as HSM servers, the mean client balking rate or mean client waiting time for various system configurations. In particular, the

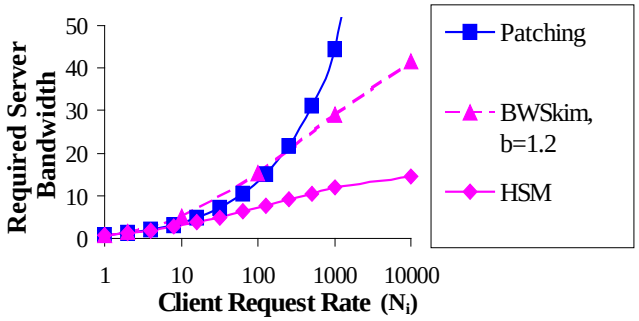


Figure 2: Required Server Bandwidth for File *i* (B_i^*)

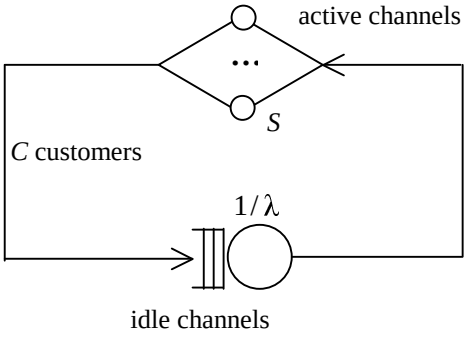


Figure 3: Client Balking Model

models can be used to determine how much server bandwidth is needed to achieve a given target client balking rate or target mean wait for a given set of media files and client access rates.

The next section develops the balking rate analysis; Section 4 develops the mean client waiting time analysis. In both cases, to enable insight across a broad spectrum of client loads and server capacities, we assume that all media files on the server are of equal (but arbitrary) play rate and duration, the request arrival process is Poisson, and each client requests the entire media file. The results derived under these assumptions can be refined for systems that have partial file accesses, media files with unequal durations or play rates, and/or other arrival processes, as noted in Section 6.

3. CLIENT BALKING RATE ANALYSIS

We consider a media server that is configured to deliver a maximum number of concurrent streams, denoted by C, each at the media play rate. If a new client request arrives when there are already C streams serving other clients, the new client "balks". That is, the client leaves the system without receiving the requested content. In the following, the unit of server capacity sufficient for delivering a single stream is called a "channel".

Below we develop a model and four alternative analytic methods for estimating the fraction of clients that balk (P_{balk}), as a function of C, the total client arrival rate, and the relative popularity of each of the media files. The solutions are validated against system simulations in Section 5.1.

3.1 Client Balking Model

The media server is similar to an $M/G/C/C$ system that has Poisson arrivals, C servers (i.e., the channels), and no additional queue for clients to wait for service, except that *service times are correlated with the number of active servers* (i.e., stream durations that are correlated with the number of active channels). Thus, the system can be represented by the closed two-center queueing network with C customers depicted in Figure 3, in which customers at the queueing center represent idle channels that are waiting to serve a new client request while customers at the upper delay center represent active channels. The queueing center generates new active channels at rate equal to the client request rate λ as long as at least one channel is idle. The service times at the delay center (i.e., the active stream durations) depend on the streaming protocol and are correlated with the number of active streams, with overall mean denoted by S. Once a stream ends, the channel joins the idle channel pool in the lower queue, waiting to be reactivated by a new client request. Note that the clients that arrive (at rate λ) when the queue of idle channels is empty depart immediately (i.e., balk) and have no impact on the operation of the channels modeled in Figure 3.

Input	Definition
C	server capacity or bandwidth, in units of the media play rate
λ	average total client request rate
K	number of files available on the server
α	parameter of Zipf file access distribution
T	file play duration
Output	
P_{balk}	fraction of clients that balk, balking rate
p_i	fraction of requests for file i, $p_i = (1/i^{\alpha})/\Sigma(1/j^{\alpha})$
N_i	normalized client request rate for file i, $N_i = p_i \lambda T_i$
$B_{i,p}^*$	required server bandwidth for file i using protocol p
S	overall mean stream duration
X	system throughput, $X = \lambda (1 - P_{balk})$
Q	mean number of idle server channels
R	mean queueing center residence time
U	average fraction of time a server channel is busy

Table 1: Notation for Client Balking Model

Two features of the model make the solution significantly more challenging than a queueing system with general independent service times at the delay center. First, the average stream duration, S, is a function of system throughput, X, which is an output of the analysis. Second, the service time at the delay center is correlated with the number of active streams (i.e., with the number of customers in the delay center). Since the dependence between stream duration and number of active streams is difficult to characterize, we first develop solutions that ignore the correlation, and then discover a solution that approximately captures the impact of the correlation.

The notation for the balking model in Figure 3 is defined in Table 1. Due to the one-to-one correspondence between channel activations and non-balking client requests, the system throughput, denoted by $X = \lambda(1-P_{balk})$, can be interpreted as the channel activation rate. Thus, by Little's law [19] the following equation relates the 2-center system measures to the measure of interest, namely, the balking rate:

$$\frac{C}{R+S} = X = \lambda(1 - P_{balk}) . \quad (4)$$

A second observation that enables the analysis is that S can be estimated from the (unknown) throughput of the queue, using Little's law, as

$$S \approx \frac{\sum_{i=1}^{K} B_{i,p}^{*}(p_i XT)}{X}, \quad (5)$$

where $B_{i,p}^*$ is the required server bandwidth computed using equation (1), (2) or (3) with N_i replaced by $p_i XT$. This is an approximation since equations (1) – (3) were derived assuming Poisson stream initiations, which is not the case for the system with a finite number of channels.

Equations (4) and (5) provide three equations that relate four unknowns, namely: X, S, R, and P_{balk}. Below we propose four methods for obtaining further equations for solving the model. The solutions in Sections 3.2 and 3.3 are motivated by simplicity, the solution in Section 3.4 is motivated by refinements that should improve the accuracy of the first two models, and the model in Section 3.5 includes a further refinement to the model in 3.4 that partially captures the correlation between expected stream duration and the number of active streams when a client initiates a new stream. Section 5 will show that this fourth model has the best overall accuracy for estimating client balking rate in patching, HSM, and bandwidth skimming systems.

3.2 *LIS–AMVA* Solution

One of the simplest solutions to the model in Figure 3 is to assume that stream durations are *load independent* (i.e., not correlated with the number of active streams) and to solve the resulting machine repair model (with unknown S) using Schweitzer's approximation [1, 2, 21] to estimate the mean time that a channel remains idle (i.e., the mean residence time at the queueing center):

$$R \approx \frac{1}{\lambda}(1+\frac{(C-1)}{C}XR)\,, \tag{6}$$

where, using Little's law, the mean queue length at the queueing center, Q, has been replaced by XR.

Equations (4), (5) and (6) define the *LIS–AMVA* model that can be iteratively solved by successive substitution or some other suitable technique for solving a system of non-linear equations.

Two sources of error in the above *LIS–AMVA* solution motivate consideration of further solutions. The first is that balking rate may be inherently difficult to estimate. In particular, a small percentage error in X, as might result from the Schweitzer approximation in (6), could yield a large error in P_{balk} as calculated from (4). A second source of error is the assumption that average time at the delay center is independent of the number of customers at the delay center.

3.3 *LIS–IC* Solution

This next approach assumes that service times at the delay center are load independent, but does not involve estimating mean residence time at the queueing center. Instead, we directly estimate the probability that all channels are busy, which is the balking rate, P_{balk}. A very simple approximation that the states of the channels are independent, yields

$$P_{balk} \approx U^C\,. \tag{7}$$

Using $U = S/(R+S)$ and the first equality in (4) yields

$$UC = \frac{SC}{R+S} = SX\,, \tag{8}$$

which, by (5), can be re-written as

$$U \approx \frac{1}{C}\sum_{i=1}^{K} B^*_{i,p}(p_i XT)\,. \tag{9}$$

Finally, the second equality in (4) together with (7) gives

$$X \approx \lambda(1-U^C)\,. \tag{10}$$

Equations (8) – (10) can be iteratively solved to obtain the estimated balking rate, U^C. Although it seems that the assumption of independent channel states could be quite inaccurate, validation results discussed in Section 5.1 show that the *LIS–IC* solution provides more accurate results than the *LIS–AMVA* solution for HSM systems.

3.4 *LIS–EL/CMVA* Solution

A third solution is motivated either by eliminating the Schweitzer approximation used in the *LIS–AMVA* model or by a desire to improve the estimate of balking rate used in the *LIS–IC* solution. In either case, note that if the service times at the delay center are assumed to be load-independent with known mean S, then the model in Figure 3 is a *separable* queueing network [19] that is equivalent to an ordinary $M/G/C/C$ queue with arrival rate λ and mean service time S.

To improve the estimate of balking rate, note that by the well-known BCMP result [4], the separable network has the same solution as the $M/M/C/C$ queue. That is, the solution for generally distributed delay center service times with mean S is the same as for exponentially distributed service times with mean S. The balking rate is equal to the probability that the queue contains C customers, which is given by Erlang's loss (EL) formula for the $M/M/C/C$ queue [18]:

$$P_{balk} \approx \frac{(\lambda S)^C / C!}{\sum_{i=0}^{C}(\lambda S)^i / i!}\,. \tag{11}$$

Substituting the above into equation (4) yields

$$X \approx \lambda\left(1-\frac{(\lambda S)^C / C!}{\sum_{i=0}^{C}(\lambda S)^i / i!}\right). \tag{12}$$

Equations (12) and (5) can be solved iteratively to estimate S and X, and then equation (11) can be applied to obtain the balking rate for this *LIS–EL* solution.

Equivalently, we can solve the model by estimating the mean residence time at the queueing center using the arrival instant theorem for the number of customers in the queue at the arrival instant [19], rather than Schweitzer's approximation. Let $X(c)$, $R(c)$, and $Q(c)$, respectively, denote the throughput, mean residence time at the queueing center, and mean queue length at the queueing center, when the network contains c customers, c = 1, 2,…,C. Using the arrival instant theorem and equation (5) with $X=X(C)$, the remaining set of equations for this *LIS–CMVA* ("customized MVA") solution are:

$$R(c) \approx \frac{1}{\lambda}(1+Q(c-1)) \tag{13a}$$

$$X(c) = \frac{c}{R(c)+S}\,,\ c = 1, \ldots, C \tag{13b}$$

$$Q(c) = X(c)R(c)\,, \tag{13c}$$

with the boundary condition $Q(0) = 0$. To solve the system numerically, S can be initialized to some suitable value, and then equations (13) followed by equation (5) with the new estimate of $X = X(C)$ can be repeatedly applied until convergence.

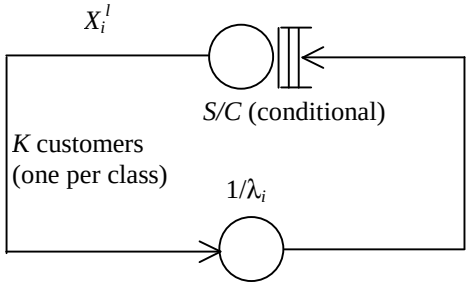


Figure 4: Waiting Model for Lead Client Requests

3.5 *CMVA* Solution

The fourth solution method is derived from the observation that in the computational structure shown in (13a – c), the same mean stream duration, *S*, is used for every network population, whereas mean stream duration should increase when the network population (or average number of active streams) decreases.

The *CMVA* solution modifies equation (5) to estimate the expected stream duration for each network population, as follows:

$$S(c) \approx \frac{\sum_{i=1}^{K} B_{i,p}^{*}(p_i X(c) T)}{X(c)}, \quad c = 2, \ldots, C, \tag{14}$$

and $S(1) = T$. In this approach, an iterative solution of equation (14) and equations (13a – c) with *S* replaced by $S(c)$ in (13b) is required at each population level, *c*, in order to determine the value of $S(c)$, which approximately captures the correlation between mean stream duration and the number of active streams.

4. CLIENT WAITING TIME ANALYSIS

We again consider a media server that is configured to deliver a maximum of *C* streams concurrently. For the models developed in this section, the clients that arrive when there are already *C* active streams wait to be served. The first request to wait for any particular media file will wait in a first-come-first-served (FCFS) queue for server bandwidth to become available. If an arriving request finds at least one request for the same file already waiting in the queue, the new request batches with the request(s) already waiting. Thus, the maximum number of request groups waiting for service is equal to the number of files on the server.

The models developed in this section assume that while a client waits in the queue for a channel that will deliver the beginning of the file, the client does not listen to and buffer any other on-going stream that may be serving the same file. Section 5.3 provides simulation results that show the impact of listening while waiting on mean client wait.

As in systems with client balking, stream duration is correlated with the number of active streams for the file, and the overall mean stream duration, *S*, is a function of system throughput, which is an output of the analysis. Furthermore, clients batching together while waiting in the queue presents another challenge to developing a simple, accurate, and efficient analytic method for computing the mean client wait. To develop the requisite new models, we again proceed initially by ignoring the correlations between stream durations and the number of active streams.

Output	Definition
W_i	mean wait for file *i* requests
W_i^l	mean wait for file *i* lead requests
Q_i^l	mean queue length of file *i* lead requests ($0 \le Q_i^l < 1$)
X_i^l	throughput of file *i* lead requests
X^l	total throughput of lead requests

Table 2: Additional Notation for Waiting Model

4.1 Basic Mean Wait Analysis

We first estimate the mean waiting time for the "lead requests" that arrive for each stream. These lead requests are either served immediately (if the server has unused capacity when the lead request arrives) or they are the first request to wait for a given new stream. The mean wait for other requests that batch with the lead requests will be estimated from the mean wait of the lead requests, as discussed below.

The initiations of new streams by the lead requests are modeled with a two-center queueing network that has *K* customer classes (one per file) with each class having a single customer, as illustrated in Figure 4. The lower delay center in the network models the mean time until a new lead request for file *i* arrives, which is equal to the inverse of the file *i* request arrival rate. The upper FCFS queue models the time that the lead request waits in the server queue for an available server channel. As soon as a lead request obtains a server channel, the customer moves to the lower delay center to generating a new lead request. If the FCFS queue is empty and at least one server channel is unused when a lead request arrives, then the service time in the FCFS queue is zero. While the FCFS queue is not empty, server channels become available to serve waiting lead requests on average every *S*/*C* units of time, where *S* is the mean stream duration.

Using the notation defined in Tables 1 and 2, the mean residence time of a lead request for file *i* at the FCFS queue in Figure 4, or the mean wait for file *i* lead requests, can be approximated as

$$W_i^l \approx \frac{S}{C}\left(U^C + \sum_{\substack{j=1 \\ j \neq i}}^{K} Q_j^l\right). \tag{15}$$

Here U^C is used to approximate the probability that a lead request finds all channels busy, where *U* is the average channel utilization. Note that each lead request that is ahead of the file *i* lead request in the queue contributes *S*/*C* to the lead request's mean waiting time.

S can be approximated by

$$S \approx \frac{\sum_{i=1}^{K} B_{i,p}^{*}(X_i^l T)}{X^l}, \tag{16}$$

where $X_i^l T$ is the average number of new streams initiated per time *T* and X^l is the total throughput of the lead requests.

U is given by

$$U = \frac{SX^l}{C}, \tag{17}$$

and Q_i^l can be obtained from Little's Law as

$$Q_i^l = X_i^l W_i^l . \quad (18)$$

Note that, for any incoming request, Q_i^l can be interpreted as the probability that there is already a lead request for file *i* waiting in the queue. Since a request for file *i* batches with any other request waiting for that file, X_i^l and Q_i^l must satisfy

$$X_i^l = (1 - Q_i^l)\lambda_i . \quad (19)$$

Substituting equation (18) for Q_i^l into (19) gives the following expression that can be derived directly from Figure 4:

$$X_i^l = \frac{1}{W_i^l + 1/\lambda_i} . \quad (20)$$

Either (20) or (19) together with equations (15) – (18) yields a system of non-linear equations that can solved iteratively to compute W_i^l.

The average number of file *i* requests that batch with a lead request for file *i* is equal to $\lambda_i W_i^l$. The mean waiting time of these requests that batch with leading requests depends on the distribution of the waiting times of the lead requests, which is not computed in the analysis above. Due in part to the upper bound on the wait queue length, one might expect the waiting time distribution to have low variance. If the lead request waiting times are assumed to be (approximately) deterministic, then the average waiting time of requests that batch with a lead request is $W_i^l/2$, and the overall mean waiting time for file *i* requests is

$$W_i = \frac{\lambda_i W_i^l \frac{W_i^l}{2} + W_i^l}{1 + \lambda_i W_i^l} . \quad (21)$$

Although the waiting times for lead requests are not deterministic, validations presented in Section 5.1 show that equations (15) – (19) and (21) are sufficiently accurate for the HSM and bandwidth skimming systems, over a wide range of system configurations and client loads.

4.2 Improved Analysis for Patching

For patching, we first note an issue with respect to choice of optimal threshold values. Unlike in [6, 14, 22], it is assumed implicitly in expresssion (16) above, and in our simulations in Section 5, that the threshold for a file *i* is computed based on the throughput of lead requests for the file (i.e., on the rate of stream initiations), which may be lower than the arrival rate of requests for the file (many of which may batch together in the queue). This policy can substantially improve performance, and we have observed no case in which it degrades performance. We also note that queueing can substantially impact patching, since by the time a new stream is initiated for a queued request, any previous stream serving the same file may have passed the threshold.

Simulation results discussed in Section 5.1 indicate that the basic mean wait model above (equations (15) – (19) and (21)) underestimates the mean waiting time in patching systems (e.g., see Figure 10). The simulations further show that the queueing of client requests decreases the burstiness of new stream initiations. A derivation of the required server bandwidth for patching in the case of *deterministic* request interrarrival times is given in the Appendix. Let Q^l denote the total mean queue length of lead requests as predicted by equations (15) – (19), and let $\hat{W}_i^l$ denote the mean wait of a file *i* lead request assuming deterministic stream initiations. That is, $\hat{W}_i^l$ is computed using equations (15) – (19) with $B_{i,patching}^*$ in equation (16) replaced by $\hat{B}_{i,patching}^*$ from equation (A.1) in the Appendix. We have experimentally derived the following heuristic interpolation that more accurately models the mean wait for lead requests in a patching system:

$$W_{i\ interpolation}^l = \left(1 - \sqrt{\frac{Q^l}{K}}\right) W_i^l + \sqrt{\frac{Q^l}{K}} \hat{W}_i^l , \quad (22)$$

which indicates that the arrival of lead requests to the server is less bursty as the total mean queue length increases. The improved estimate of overall mean wait for file *i* requests in a patching system is then computed using equation (21), with the mean wait for lead requests computed using equations (15) – (19) and (22).

The validation results in Section 5 will show that the above models of mean client wait are quite accurate. To obtain even greater model accuracy, we could consider using the arrival instant theorem for separable networks to estimate the mean queue length seen by an arriving lead request, or we could develop a more accurate approximation than U^C for the probability that a lead request finds all channels busy, as we did for estimating balking rate in Section 3. However, since mean client wait is estimated sufficiently accurately with the above simpler models, we do not consider more complex approaches further.

5. RESULTS

Section 5.1 provides comparisons between results of the analytic models developed in Sections 3 and 4, and results computed by simulating the media server with the relevant streaming protocol, over a wide range of values of the total client request rate ($N = \lambda T$), number of popular media files on the server (K), and server bandwidth capacity (C). The simulations include Poisson request arrivals, the Zipf distribution of file access frequencies, stream initiations when server bandwidth is available, stream terminations and client merges dictated by the streaming protocol, and client balking or queueing when there are C active streams. The validation results will show that for each protocol, the most accurate analytic estimates of client balking rate and mean client waiting time can be used to evaluate protocol performance for system configurations of practical interest.

Following the model accuracy assessments, Sections 5.2 through 5.4 provide new evaluations of the protocols with respect to the client QoS measures. Section 5.2 considers a potential enhancement for the protocols. Section 5.3 evaluates the protocols with respect to the server bandwidth needed to achieve a given target (low) client balking rate or target (low) mean waiting time. Section 5.4 compares the policies with respect to how well a system configured for a given anticipated client load performs under small to moderate increases in the actual load on the system.

For simplicity, all of the experiments in this section assume the relative file access frequencies are given by the Zipf distribution with parameter α equal to one (see Table 1). Results could as easily be obtained for other values of α or for other distributions of relative file access frequencies.

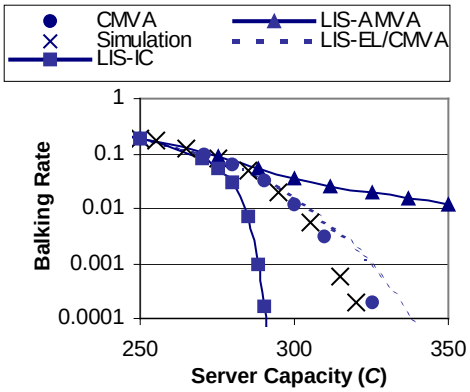
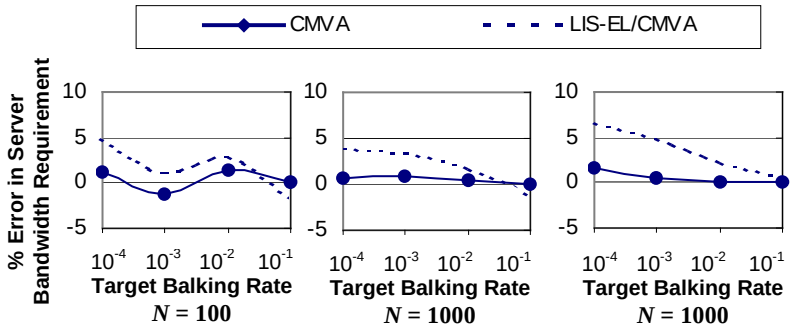
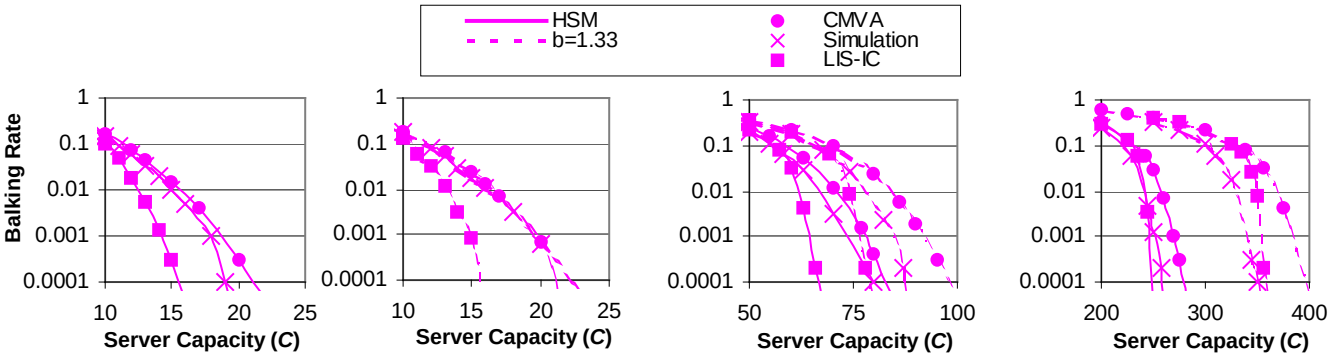


Figure 5: Example Balking Model Results for Patching (100 Files, N = 1000)



(a) 100 Files **(b) 20 Files**

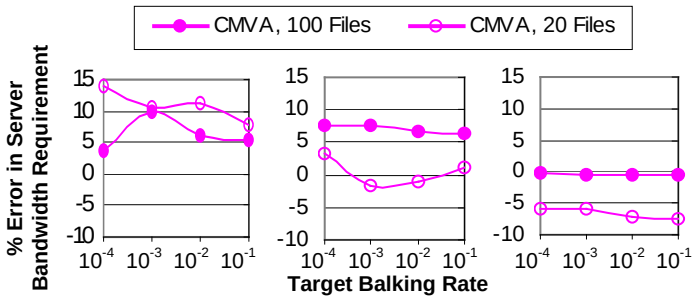
Figure 6: Balking Model Accuracy for Patching Systems



HSM **b = 1.33**

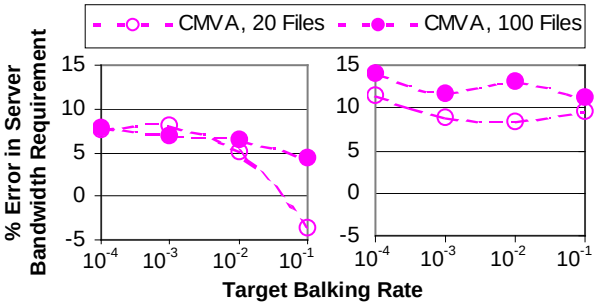
(a) N = 10 **(b) N = 100** **(c) N = 1000**

Figure 7: Example Balking Model Results for HSM and Bandwidth Skimming Systems (100 Files)



(a) N = 100 **(b) N = 1000** **(c) N = 10000**

Figure 8: *CMVA* Accuracy for HSM Systems



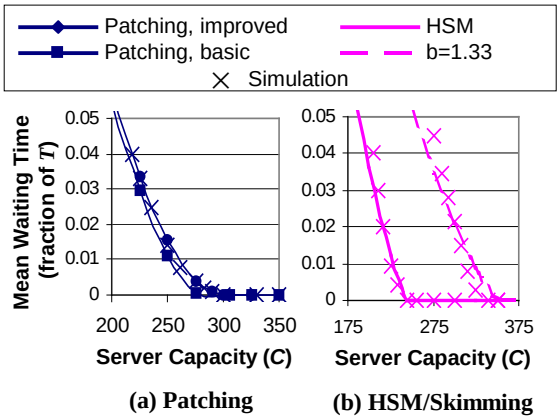
(a) N = 100 **(b) N = 1000**

Figure 9: *CMVA* Accuracy for Bandwidth Skimming

5.1 Analytic Model Validations

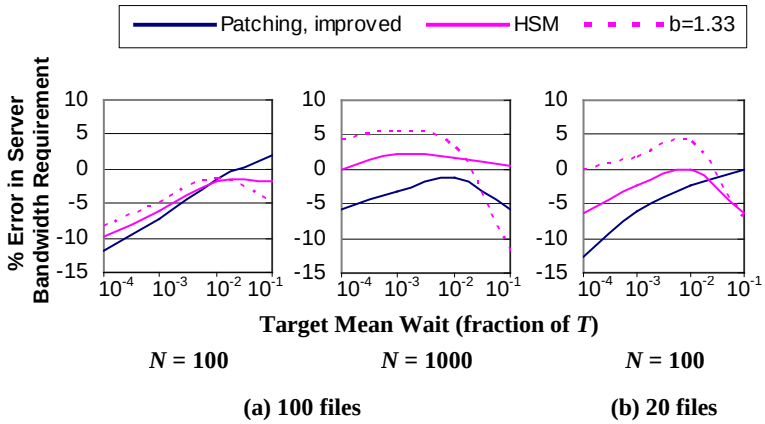
Figure 5 provides example results of client balking rate as a function of the media server bandwidth, C, estimated by system simulation and by the four analytic methods developed in Sections 3.2 – 3.5, for a patching system with K = 100 popular media files and total client request rate, N, equal to an average of 1000 client requests per period of time equal to the media playback duration T. Note that the *CMVA* estimates agree extremely well with the simulation results. The *LIS–AMVA* estimates are less accurate than the *LIS–CMVA* estimates due to the high sensitivity of the balking rate measure to small errors in the system throughput estimate. Figure 6 shows the percent error in each of the two most accurate analytic estimates, compared with simulation estimates of the server capacity needed as a function of the target balking rate. Note that for 20 or 100 files, average total client request rates of 100 or 1000, and target balking rate varying from one in ten to one in ten thousand, the *CMVA* estimates are within 2% of the simulation estimates for the patching systems. Similar accuracy of the *CMVA* estimates was also observed for higher client loads.

Figure 7 provides example validation results, and Figures 8 – 9 summarize the percent error for the most accurate analytic estimates of balking rate in HSM systems and in bandwidth skimming systems with b = 1.33. Note that for 20 or 100 files and



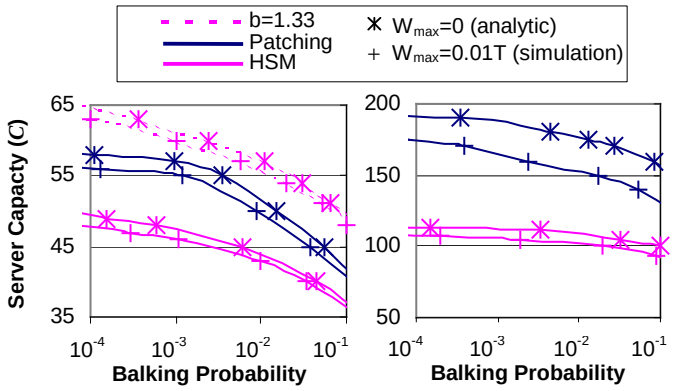
(a) Patching (b) HSM/Skimming

Figure 10: Example Waiting Model Results (100 files, N = 1000)



(a) 100 files (b) 20 files

Figure 11: Waiting Model Accuracy



(a) N = 100 (b) N = 1000

Figure 12: Impact of Non-Zero Maximum Wait (20 files)

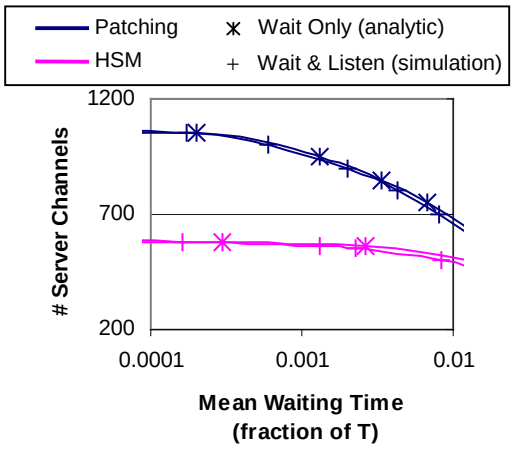


Figure 13: Impact of Listen While Waiting (100 files, N = 10000)

total client load varying from 100 to 10,000 requests, on average, per time T, the *CMVA* estimates of the HSM server bandwidth needed for a given target balking rate have 0 – 15% absolute error. For 20 files, the percent error in the *CMVA* estimates is slightly lower than for 100 files for large N; thus, at extremely high client request rate (e.g., N = 10,000) for 20 files, the *CMVA* solution slightly underestimates (i.e., by 5 – 7%) the HSM server capacity needed to achieve the target balking rate. For less extreme client load or a larger number of popular files, the *CMVA* estimates are instead slightly pessimistic (i.e., overestimate required server bandwidth for a given desired balking rate) for the HSM system. Figure 9 shows that the *CMVA* model is slightly pessimistic for the bandwidth skimming (b = 1.33) systems as well.

Figure 10 provides example validations, and Figure 11 summarizes the accuracy of the mean client waiting time models for each of the three protocols. The basic model for HSM and bandwidth skimming, and the improved model for patching, estimate the server bandwidth needed to achieve the target mean client wait within 15% absolute error compared with the simulation estimates. Note also that in the range of practical target client waiting times (i.e., 0.001T to 0.01T), and for a client request rate for the collection of popular files significantly greater than 10, the analytic estimates are within 5% of the simulation values for all three protocols. Furthermore, the analytic estimates are less optimistic for the HSM systems than for the patching systems. Hence, any analytically estimated performance gains for HSM as compared with patching will be conservative.

Figure 12 considers the case in which clients are willing to wait for a short time before balking. Specifically, if the wait exceeds W_{max}, they balk. We obtain the results for W_{max} = 0.01T using simulation. As shown in the figure, the server bandwidth needed to achieve a given balking rate is similar for W_{max} = 0.01T and W_{max} = 0, except in patching systems with high N, where HSM significantly outperforms patching. Thus, the analytical estimates of server capacity needed for a given target balking rate when W_{max} = 0 are just very slightly conservative for systems where clients balk after a short waiting time.

For a large fraction of the system configurations we simulated, we found the balking rate estimate from a simulation run length that has 1000 balking events to be essentially identical to the estimate from a run length that has only 100 balking events. Thus, simulating more than 1000 balking events would yield negligible improvement in accuracy. The corresponding total numbers of client request arrivals also yielded essentially identical estimates of mean client wait. These observations were used in determining simulation run length. Although a few curves have odd

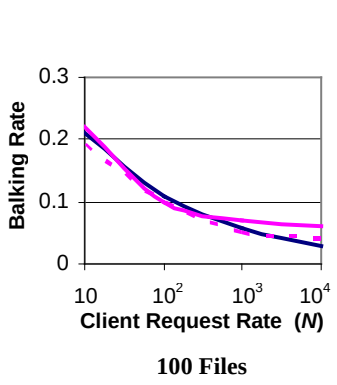
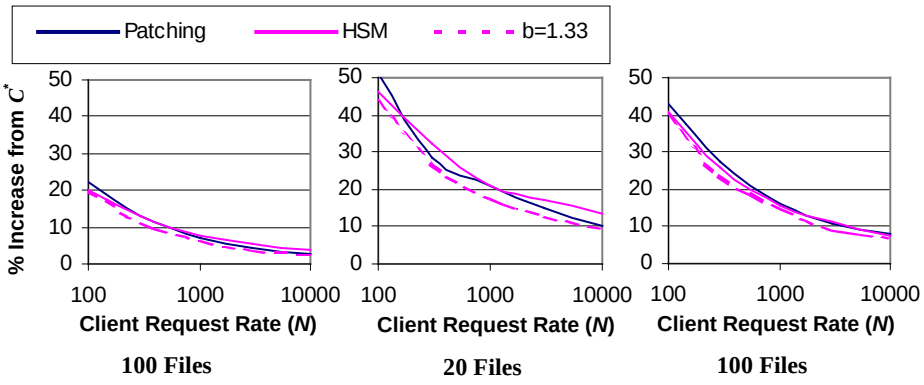


Figure 14: Balking Rate at $C = C^* = \sum B^*_{i,p}(N_i)$



(a) Target Balking Rate= 0.01 **(b) Target Balking Rate= 0.0001**

Figure 15: Server Bandwidth Needed for Low Target Balking Rate (% increase from $C^* = \sum B^*_{i,p}(N_i)$ **)**

"wiggles", we have verified that these are not artifacts of the run length selection.

It takes on the order of 20-50 minutes to simulate 1000 balking events for a single system configuration with moderate to high client request rate and low balking rate. Thus, the highly efficient and accurate analytic estimates offer a key advantage for rapidly exploring system design issues such as those explored below.

5.2 Possible Protocol Enhancement

For the mean waiting time results presented above and estimated by the analytical models, clients waiting for service do not listen to any on-going stream that may be serving the same file. As shown by the simulation results in Figure 13, listening while waiting has approximately the same performance for either the patching or the HSM systems considered in this work.

The lack of significant performance improvement for listening while waiting may be explained by two factors. First, the required length of a new stream under a "listen while waiting" strategy is determined by when the *last* request arrived in that batch. Second, "listen while waiting" offers significant improvements only if the time spent in the queue is relatively substantial, yet practical systems are designed for short mean client wait.

5.3 Server Bandwidth Needed for Target QoS

In this section we address the questions (a) what is the mean waiting time, or the client balking rate, when the server is configured with the required server bandwidth defined in Section 2 for a given set of popular media objects and a given client load, and (b) how much server bandwidth is needed to achieve a given target client balking rate, such as one in ten thousand, or a given target mean waiting time, such as 0.001*T*, where *T* is the requested media playing time. Figures 14 and 15 provide answers to these questions for client balking rate; Figures 16 and 17 provide answers for mean client wait, over a wide range of client loads.

In Figures 14 and 16, at each total client request rate (*N*), the media server using a given protocol has bandwidth capacity equal to the sum of the required server bandwidth as given in Section 2 for each file *i* (i.e., for protocol *p* $C = C^*(N,K,\alpha) = \sum_{i=1}^{K} B^*_{i,p}(N_i)$).

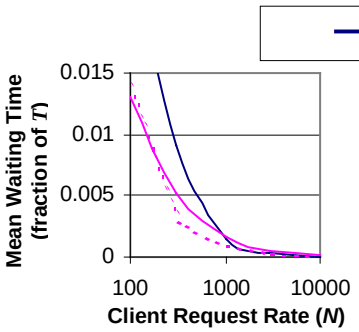
Figure 14 shows that when the server is configured in this way, for patching, HSM, or bandwidth skimming, the client balking rate ranges from over 20% to 5 – 10% as client request rate ranges from 10 to 10,000 requests per time *T*. The results are similar if the server contains 20 media files (not shown). Such balking rates are unacceptably high for most systems, but Figures 15(a) and (b) show that only a modest increase in server bandwidth is needed to achieve a balking rate as low as one in one hundred or as low as one in ten thousand, respectively. For example, if the server contains 100 popular files and the total client request rate for the popular files is 1000 requests per time *T*, then for each of the three protocols, server bandwidth equal to approximately 1.15*C** achieves a balking rate of one in ten thousand.

Figure 16 shows that when the server is configured with bandwidth equal to $C^*(N,K,\alpha)$ and the total request rate *N* is greater than 100, the mean client wait is a small fraction of *T*. Figure 17 shows further that only small increases in server bandwidth are needed to achieve very low mean client wait.

The small changes in *C** needed to achieve a low balking rate or low target mean wait illustrate the advantage of aggregating the client load for many popular files. During periods when one media file needs more than its average server bandwidth, another file may use less than its average server bandwidth. Thus, the variation in the sum of the server bandwidths needed to give every client immediate service is statistically lower than the variation for each file. A similar result is shown for mean waiting time over a much smaller range of system configurations in [12].

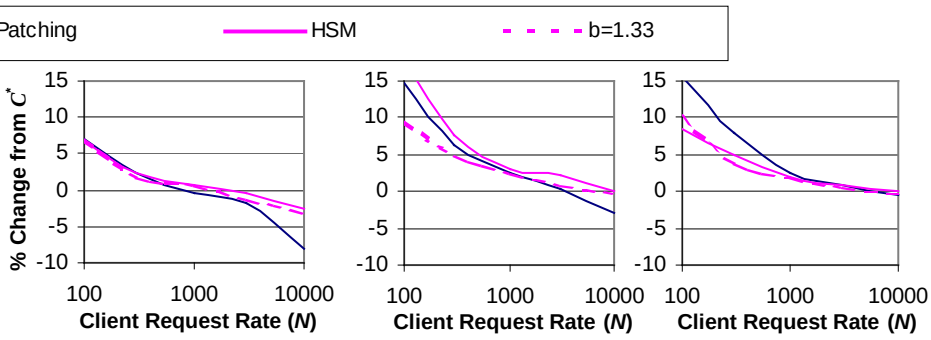
Although the value of *C** is different for each streaming protocol, Figures 15 and 17 show that the percent increase in *C** needed to achieve a given target balking rate is similar for all three protocols.

Note that as *N* increases, the required server bandwidth per client decreases (due to the sublinear growth in $B^*_{i,p}$ illustrated in Figure 1) *and* the percent increase in *C** needed to achieve a given low balking rate or mean wait also decreases (as shown in Figures 15 and 17). This illustrates the significant advantages of serving larger client populations from a given scalable media server.



100 Files

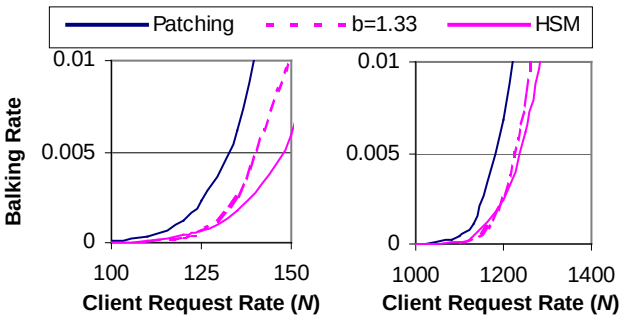
Figure 16: Mean Wait at $C = C^* = \sum B^*_{i,p}(N_i)$



100 Files | 20 Files | 100 Files

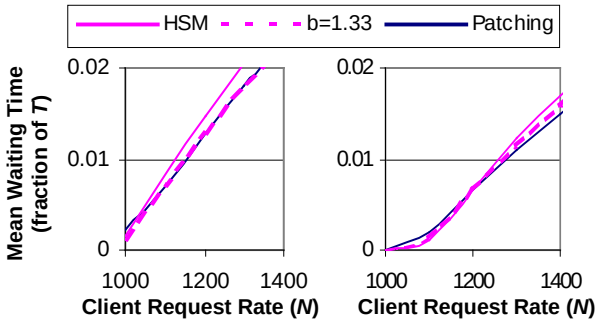
(a) Target Wait = 0.001*T* | (b) Target Wait = 0.0001*T*

Figure 17: Server Bandwidth Needed for Low Target Mean Client Wait (% change from $C^* = \sum B^*_{i,p}(N_i)$)



(a) Anticipated N = 100, 20 Files ($C = 1.5C^*_{N=100}$) | (b) Anticipated N = 1000, 100 Files ($C = 1.2C^*_{N=1000}$)

Figure 18: Balking System Robustness (Target Balking Rate = 0.0001 at anticipated *N*)



(a) Target $W = 0.001T$ ($C = C^*_{N=1000}$) | (b) Target $W = 0.0001T$ ($C = 1.05C^*_{N=1000}$)

Figure 19: Waiting System Robustness (100 files, Anticipated N = 1000)

5.4 System Robustness

Figures 18(a) and (b) and 19(a) and (b) each consider the case in which a scalable media server is configured with bandwidth equal to the bandwidth needed for a given anticipated client load (N) and a given low target balking rate or mean wait, respectively. For example, in Figure 18(a) the server is configured with the bandwidth needed for client request rate N = 100 and balking rate equal to one in ten thousand. As shown in Figure 15(b), C = 1.5C* for this target balking rate, 20 files, and N = 100. For each such system, the client request rate is varied up to 1.4 – 1.5 times the anticipated load, to determine how much the system performance degrades when the actual client load is somewhat higher than anticipated.

Figures 18(a) and (b) show that the HSM and bandwidth skimming systems can tolerate a 10% increase in client request rate without significant degradation in client balking rate, but system performance degrades rapidly for increases in client load greater than 10%. Patching systems are somewhat more sensitive to client load increases, tolerating closer to a 5% increase before service begins to rapidly degrade.

Figures 19(a) and (b) show that mean waiting time is only insensitive to increases up to 10% in client load if the server is configured for very low target mean wait (e.g., target mean wait equal to 0.0001T). Mean wait degrades somewhat less rapidly than balking rate, but performance still degrades significantly for greater than 10% increases in client load.

6. CONCLUSIONS

This paper has developed simple, highly accurate and efficient analytic estimates of mean client wait and the fraction of clients that balk when a scalable on-demand media server is configured to deliver a given maximum number of streams concurrently. The customized AMVA estimates are noteworthy in that they capture complex phenomenon that would suggest that more complex state space analyses would be needed. In particular, the balking rate analysis uses customized AMVA to estimate a probabilistic measure rather than the usual "mean value" measures, suggesting that customized AMVA techniques may be applicable to a significantly broader range of performance metrics and system design questions than has been previously demonstrated.

The results in Section 5 show that (a) scalable media servers that are configured with the "required server bandwidth" defined in previous work have low mean wait but may have unacceptably high client balking rates (i.e., greater than one in twenty), (b) for high to moderate client load, a 10 – 50% increase in the previously defined required server bandwidth will achieve a very low balking

rate (i.e., one in ten thousand), and (c) media server performance (either mean wait or balking rate) degrades rapidly if the actual client load is more than 10% greater than the anticipated load. The analytic models can be easily applied to determine the server bandwidth needed for a given number of media files, anticipated total client request rate and file access frequencies, and target balking rate or mean wait. The results can also be refined for systems with media files that have unequal play rates or durations, client arrivals that are more bursty than the Poisson process, and partial file requests, either by modifying the simulator or by extending the analytic models to include these system features.

ACKNOWLEDGEMENTS

We thank Ahmed Ayad, Babis Samios, and the anonymous Sigmetrics 2002 reviewers for helpful comments on this work.

REFERENCES

[1] Y. Bard, "A Model of Shared DASD and Multipathing", *Comm. ACM* 23, 10 (Oct. 1980), pp. 564-572.

[2] Y. Bard, "A Simple Approach to System Modeling", *Performance Evaluation* 1, 3 (Aug. 1981), pp. 225-248.

[3] 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.

[4] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *J. ACM* 22, 2 (Apr. 1975), pp. 248-260.

[5] S. Carter and D. Long, "Improving Video-on-Demand Server Efficiency Through Stream Tapping", *Proc. ICCCN '97*, Las Vegas, NV, Sept. 1997.

[6] Y. Cai, K. A. Hua, and K. Vu, "Optimizing Patching Performance", *Proc. MMCN '99*, San Jose, CA, Jan. 1999.

[7] 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.

[8] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley, "Channel Allocation under Batching and VCR Control in Video-on-Demand Systems", *J. Parallel and Distributed Computing* 30, 2 (Nov. 1995), pp. 168-179.

[9] D. L. Eager and M. K. Vernon, "Dynamic Skyscraper Broadcasts for Video-on-Demand", *Proc. MIS '98*, Istanbul, Turkey, Sept. 1998.

[10] 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.

[11] 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), pp. 742-757.

[12] 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.

[13] L. Gao, J. Kurose, and D. Towsley, "Efficient Schemes for Broadcasting Popular Videos", *Proc. NOSSDAV '98*, Cambridge, UK, July 1998.

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

[15] A. Hu, "Video-on-Demand Broadcasting Protocols: A Comprehensive Study", *Proc. IEEE Infocom 2001*, Anchorage, AL, Apr. 2001.

[16] 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.

[17] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems", *Proc. ACM SIGCOMM '97*, Cannes, Sept. 1997.

[18] L. Kleinrock, *Queueing Systems Volume 1: Theory*, John Wiley and Sons, New York, NY, 1975.

[19] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

[20] J. F. Paris, S. W. Carter, and D. D. E. Long, "A Hybrid Broadcasting Protocol for Video on Demand", *Proc. MMCN '99*, San Jose, CA, Jan. 1999.

[21] P. Schweitzer, "Approximate Analysis of Multiclass Closed Networks of Queues", *International Conference on Stochastic Control and Optimization*, Amsterdam, Netherlands, 1979.

[22] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal Patching Schemes for Efficient Multimedia Streaming", *Proc. NOSSDAV '99*, Basking Ridge, NJ, June 1999.

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

APPENDIX

This appendix derives the required server bandwidth for immediate on-demand streaming using the patching protocol and for a given media file with deterministic request interarrival times.

Let T, y, and λ_i denote the play duration, patching threshold (as a fraction of T), and request rate, respectively, for any given media file i. Since request interarrival times are deterministic, for each full-file multicast there will be precisely $m = \lfloor \lambda_i y T \rfloor$ patch streams initiated, and the duration of these patch streams, in their initiation order, will be $1/\lambda_i$, $2/\lambda_i$,…, m/λ_i. The elapsed time between successive full-file multicasts is simply $(m+1)/\lambda_i$. Therefore, the required server bandwidth for file i is given by

$$\hat{B}^*_{i,patching} = \frac{T + \sum_{j=1}^{m} \frac{j}{\lambda_i}}{(m+1)/\lambda_i} = \frac{2N_i + m(m+1)}{2(m+1)}, \qquad \text{(A.1)}$$

where N_i is the average number of requests that arrive for the file per period of length T. By taking the derivative with respect to m and setting the result to zero, $\sqrt{2N_i} - 1$ is found to be the (possibly non-integral) value of m that minimizes required server bandwidth. The integral value of m that minimizes required server bandwidth is either $\lfloor \sqrt{2N} - 1 \rfloor$, or $\lceil \sqrt{2N} - 1 \rceil$. Substitution of each of these expressions into (A.1) above, and taking the minimum, yields the required server bandwidth with optimal threshold. Note that the required server bandwidth grows with the square root of the request rate, as in the case of Poisson arrivals.