RC 20038 (88607) 04/17/95
Computer Science

IBM Research Report

The Role of Wait Tolerance in Effective Batching: A Paradigm for Multimedia Scheduling Schemes

Hadas Shachnai and Philip S. Yu

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

The Role of Wait Tolerance in Effective Batching: A Paradigm for Multimedia Scheduling Schemes

Hadas Shachnai
Philip S. Yu
IBM TJ Watson Research Center
Yorktown Heights, NY 10598

In a video-on-demand (VOD) environment, batching requests for the same video to share a common video stream can lead to significant improvement in throughput. Using the *wait tolerance* characteristic that is commonly observed in viewers behavior, we introduce a new paradigm for scheduling in VOD systems. We propose and analyze two classes of scheduling schemes: the Max_Batch and Min_Idle schemes, that provide two alternative ways for using a given stream capacity for effective batching. In making a video selection, the proposed schemes take into consideration the next stream completion time as well as the viewer wait tolerance. We compared the proposed schemes with the two previously studied schemes: (1) first-come-fisrt-served (FCFS) that schedules the video with the longest waiting request and (2) the maximum queue length (MQL) scheme that selects the video with the maximum number of waiting requests. We show through simulations that the proposed schemes outperform substantially FCFS and MQL in reducing the viewer turn-away probability, while maintaining a small average response time. In terms of system resources, it is found that by exploring the viewer wait tolerance, the proposed schemes can reduce significantly the server capacity required for achieving a given level of throughput and turn-away probability as compared to the FCFS and MQL. Furthermore, our study shows that an aggressive use of the viewer wait tolerance for batching may not yield the best strategy, and that other factors, such as the resulting response time and the total expected loss of viewers, should be taken into account.

Index Terms: Video-on-Demand, Scheduling, FCFS, Performance Analysis

1 Introduction

In a video-on-demand [6, 5, 8] (VOD) system, subscribers can choose both the video they wish to view and the time they wish to view it. This contrasts with services in which users can choose only from a small set of selections and watch them at pre-specified times. The server for a VOD service is essentially a homogeneous system, in which a large number of video streams of the same format, e.g., MPEG compressed, are stored and retrieved.

For a commercial environment, VOD servers may potentially need to handle thousands of requests simultaneously. There are often popular videos which are requested by many of the viewers. Batching multiple viewing requests to the same video stream can greatly increase the number of viewers supportable by a video server. In an ATM type network with multicast facility, the same video data can be sent to multiple viewers simultaneously without additional cost. Due to the isochronous requirement, each server is assumed to have a limit on the number of simultaneously supportable video streams. This is referred to as the stream capacity of the VOD server. Batching allows to support a number of viewers that is larger than the stream capacity.

In order to obtain batching we may need to delay deliberately viewer requests, so that later arrivals can share the same video stream with earlier requests. However, viewers may turn away when the *batching interval* becomes large. In general, viewers can tolerate and accept a small amount of delay (say 5 minutes). This phenomenon is referred to as *wait tolerance.* As the delay grows larger, viewers are likely to leave the system without being served. On the other hand, if the batching interval is too short, more of the system stream capacity is consumed and viewers may have to wait longer in the queue for available streams. This can also result in loss of viewers.

The conventional approach to VOD scheduling is the *first-come-first-served (FCFS)* scheme. Under this approach, all video requests join a single request queue.

Once a stream capacity becomes available, the request at the head of the queue is served. To support batching, all other requests queueing for the same video are also served by this stream. An alternative is to maintain a separate request queue for each video and select for the next showing the video with the longest queue. This is referred to as the *maximal queue length (MQL)* scheme.

As mentioned above, waiting requests may leave the queue when their waiting times exceed the tolerance of the requesting viewers. This loss of viewers is undesirable. Different scheduling schemes lead to various degrees of batching and losses of viewers. In selecting the next candidate for showing, the FCFS scheme does not take into account the batching factor of a videos (i.e the number of pending requests for specific video) while the MQL ignores the wait time already incurred by the pending requests. Comparison study of the performance of these schemes can be found in [2], where it is found that FCFS generally performs better than MQL.

In this paper we introduce a new scheduling paradigm that makes use of the viewer wait tolerance for effective batching. The central issues to be addressed here are the extent to which batching should be pursued, and the other factors that need to be considered. We propose and analyze two classes of scheduling schemes referred to as the Max_Batch and the Min_Idle schemes. Both classes of schemes produce delayed schedules that aim to increase the batching effect for popular videos. While the Max_Batch schemes maximize viewers batching and allow available streams to remain idle in the presence of pending requests, the Min_Idle schemes apply batching selectively only to more popular videos and avoid stream capacity from idling by serving unbatched requests of less popular videos, thus improving the performance. We elaborate on that in Section 3. Various stream selection criteria are also considered. We study trade-offs between increasing batching and minimizing the expected loss of viewers.

We focus on the main measure of system throughput and show (in Section 2) the computational complexity involved in finding the *optimal* strategy with respect to this measure, even for the case where we have a *complete* knowledge of system behavior. This motivated our search for efficient heuristic solutions in Section 3. A simulation program was developed to evaluate the performance of the proposed schemes. We show in Section 4 that the proposed schemes can outperform substantially FCFS and MQL in reducing the turn-away probability. A summary is given in Section 5.

We briefly comment on some related works that study alternative means to support stream sharing. In [4, 7], a bridging approach of using buffer to support stream sharing is considered. By keeping the last few minutes of a leading video request in buffer, trailing requests can be served out from the buffer. In [9], buffering is used to support pause-resume under batching in a look-ahead scheduling scheme. In [3], stream sharing is achieved through a piggybacking approach of adjusting the playout rate of the videos in progress. By slowing down the leading video request and/or speeding up the trailing video request, the two requests can be merged into one. While batching avoids the additional resource requirement as in the bridging approach or the complexity in the piggyback approach, it does cause additional waiting of the viewers. Nevertheless, batching can be used in conjunction with either the bridging approach or the piggybacking approach to strike a balance among wait time, resource requirement and implementation complexity.

2 Preliminaries

2.1 Performance Measures and basic Schemes

Assume a VOD system where the initiation of video requests corresponds to arrival process $\mathcal{A}$ with mean time $1/\lambda$ between successive requests. Upon arrival a viewer chooses with probability p_k to join the queue of video k, $1 \leq k \leq M$. The viewer waits in the queue for a certain interval of time after which he may decide to leave without being served. We define the viewer's *wait threshold* as the scheduler estimate of the general viewer tolerance on the wait time. In terms of a queuing system, the viewers are classified as customers of the *reneging* type, with a wait time given by a random variable (*rv*) W.

The system can support N_{max} simultaneously playing streams, where each stream shows one of the M videos. Whenever a stream becomes available (upon viewing completion), the scheduler inspects the queues and chooses the next video to be shown using that stream. The average length of a video is given by L.

In our analysis of scheduling schemes for the above model, we use the following performance measures:

- *Turn-away probability*: It is given by the ratio between the number of reneging viewers and the total number of arrivals in a given time interval. Minimizing the turn-away probability maximizes the system throughput, therefore in the present context we use often the term *throughput measure* when referring to minimization of the turn-away probability.

- *Average response time*: For any viewer that is served by the system (i.e. a non-reneging viewer), the time that elapses from the viewer's arrival until the start of his show is the viewer's response time. Our objective is to minimize the average response time for these viewers.

- *Fairness*: Let $q_t(i)$ denote the turn-away probability associated with the i-th video $1 \leq i \leq M$, and let q_t be the average turn-away probability of the system. We say that a system is *totally fair* if $q_t(i) = q_t, \quad \forall 1 \leq i \leq M$. As in [2], we measure the degree of unfairness in the system as the average deviation, σ_f, from q_t, using a partition of the series of requests, sorted by video ids, to $|S|$ subsets of roughly equal number of requests.
  Thus,

$$\sigma_f = \frac{\sum_{j=1}^{|S|} (q_t(i) - q_t)^2}{|S|} \tag{1}$$

For a given scheme, the system's *degree of fairness* increases as σ_f gets closer to 0.

While all of the above measures are important for achieving good quality of service, we refer in our analysis to the system throughput as the *main* measure, with the average response time and the fairness ratio the *secondary* measures. This is due to the fact, that a strategy is considered efficient if it is shown to maximize the system throughput[1]. However, some of the schemes may provide small response time with high degree of fairness and yet perform very poorly; A typical example is the *Last Come First Serve* strategy, which minimizes the average response time, while causing a large fraction of the arriving viewers to leave the system without being served (i.e. the turn-away probability is close to 1). Thus, the last two measures provide only secondary information in the overall performance evaluation of a given scheme.

2.2 The Optimal Strategy

In this section we formulate the problem of maximizing the throughput in a VOD system as a stochastic optimization problem, and derive the functional equation defining the optimal scheme. Assuming a complete knowledge of the arrival process $\mathcal{A}$, the distribution of the rv R and the request probabilities for the videos $p_1, \ldots, p_M$, our objective is to maximize the throughput in a given time interval $[0, T)$.

Typically, the scheduler is activated when a stream becomes available, i.e. *scheduling points* are identified with showing completions. Given a scheduling point t_0 and the lengths of the previously scheduled videos, we denote by $t_l,\ l \geq 1$ the time of the l-th completion after t_0. Let $l_k(t_l - t_0)$ be the expected loss of viewers for video k in the interval $[t_0, t_l)$, i.e., given the set of viewers waiting in queue k at t_0:

$$l_k(t_l - t_0) = E(\text{Departures in } (t_l - t_0)) \ .$$

[1]Note that the average response time is always bounded by the average wait time of a viewer before reneging.

Using the rv R, with a_r the arrival time of the r-th viewer in the queue of video k, we have

$$l_k(t_l - t_0) = \sum_{r=1}^{n_k(t_0)} E(R \mid a_r). \tag{2}$$

where $n_k(t_0) \equiv n_k$ is the number of viewers in the queue of video k at t_0.

A straightforward approach is the *Greedy* scheme defined as follows:

- At any scheduling point t_0, $\forall\ 1 \leq k \leq M$ compute the expected number of viewers reneging from queue k before the next completion, given by $l_k(t_1 - t_0)$, assuming k is not scheduled for showing at t_0.
- Schedule the video with maximal expected loss in the interval $[t_0, t_1)$.

It is easy to verify, that Greedy is not optimal. However, as shown in our simulation study, it provides a significant increase in the system throughput, compared to the basic MQL and FCFS schemes (We call this scheme below the BML).

Let i be the state of the system given by the waiting times of the viewers in queues $1, \ldots, M$:

$$i = (w_{1,1}, \ldots, w_{1,n_1}, w_{2,1}, \ldots, w_{2,n_2}, \ldots, w_{M,1}, \ldots, w_{M,n_M}) \ ,$$

where $w_{j,k}$ is the wait time of the j-th viewers in the queue of video k. We denote by $f(i)$ the expected loss of viewers when the system starts from state i, and the optimal strategy is used.

A decision D in state i to schedule all viewers in queue k, for some $1 \leq k \leq M$ yields the expected number of reneging viewers in the interval $[t_0, t_1)$ given by

$$r_i(D) = \sum_{m \neq k} l_m(t_1 - t_0) \ . \tag{3}$$

Let $p_{ij}(D)$ be the probability of transition to state j from state i given the decision D, then by the principle of optimality [1], if decision D is taken, and at the next scheduling point the optimal strategy f is used, then the total expected loss of viewers is

$$r_i(D) + \sum_j p_{ij}(D) f(j) \ . \tag{4}$$

Hence, the optimal strategy satisfies

$$f(i) = \min_D [r_i(D) + \sum_j p_{ij}(D) f(j)] \ . \tag{5}$$

Using successive approximations, f can be computed directly whenever the state space is finite. For the above problem, the reneging property can be used to truncate the infinite state space to a finite space for which the optimal strategy may be computed. We note that even for moderate values of M and T, the computational complexity as well as the space required for solving the functional equation for f can be enormous thus we proceed with a study of approximation schemes.

3 The Wait Tolerance Batching Approach

In this section we introduce heuristic batching schemes that use effectively the viewer wait tolerance. Intuitively, our schemes use some limited knowledge of the reneging pattern of viewers for delaying the start of a new showing based on the future completion schedule, and thus increase the batching effect in the system. We keep this delay short enough, so as to minimize the expected loss of viewers. Therefore, the overall throughput of the system increases.

We classify the M videos as *hot* or *cold* videos, based on their request frequencies. Assuming the videos are numbered in decreasing request frequencies, a classification threshold $\tau \in \{1, 2 \ldots, M\}$ determines the boundary between these two categories.

We considered the following two classes of batching schemes:

1. **Max_Batch** – A video is eligible for scheduling only if some of its requests have wait times reaching or exceeding a pre-specified *batching threshold.* This in fact

$\mathcal{H} = \{1, \ldots, M\}$.
At each scheduling point t_0 (that can either be a stream completion, a new request arrival or some delayed scheduling instant)
Let $L_{\mathcal{H}}$ denote the subset of videos in $\mathcal{H}$ with waiting requests that reached or exceeded the batch threshold by t_0.
if $L_{\mathcal{H}} \neq \emptyset$ a video is scheduled using one of the criteria:
(a) Maximal queue length (the *BMQ scheme*).
(b) Maximal expected loss till the next completion at t_1, i.e. $\max_{k \in L_{\mathcal{H}}} l_k(t_1 - t_0)$ (the *BML* scheme).
else
delay the schedule until a request reaches its batch threshold

Figure 1: The BMQ and BML schemes

imposes a *minimum wait time requirement.* If no video satisfies this eligibility criterion, an available stream remains unused and the scheduling point is delayed to a later time when a request reaches the batching threshold. The objective is clearly to maximize batching, even if that causes streams to be idling. The batching threshold is chosen based on the wait tolerance.

2. **Min_Idle Schemes** – The scheme only tries to pursue batching on hot videos without the minimum wait time requirement. Cold videos are always eligible for scheduling. The eligibility criterion is thus substantially relaxed. The objective is to provide small response time and to decrease loss of viewers among those requesting cold videos. This is done by minimizing stream idling.

The Min_Idle schemes maintain two sets of video queues, denoted by $\mathcal{H}$ and $\mathcal{C}$, while the Max_Batch schemes only maintain one set ($\mathcal{H}$) of queues. Under the Min_Idle schemes, the partition of videos to the sets $\mathcal{H}$ and $\mathcal{C}$ is defined dynamically based on the current state of the system and the classification threshold τ.

At each scheduling point t_0 (that can be either a stream completion or an arrival of a request)
  Define $\mathcal{H}$ and $\mathcal{C}$:
    A video k is in $\mathcal{H}$, if
      (a) $k \leq \tau$.
      (b) $n_k > 1$.
      (c) $n_k = 1$ and $t_0 - a_1 > \nu$;
        /* A single viewer that exceeded batch threshold */
    otherwise the video is in $\mathcal{C}$.
  Let $L_{\mathcal{H}}$ denote the subset of videos in $\mathcal{H}$ with a positive expected loss in the interval $[t_0, t_1)$, i.e. $L_{\mathcal{H}} = \{k \in \mathcal{H} \mid l_k(t_1 - t_0) > 0\}$.
  if $L_{\mathcal{H}} \neq \emptyset$ a video is scheduled using one of the
    criteria as defined for the Max_Batch case
    (*IMQ* and *IML* respectively)
  else
    schedule a video in $\mathcal{C}$ (if any) using FCFS.

Figure 2: The IMQ and IML schemes

3.1 The Max_Batch schemes

The Max_Batch schemes are presented in Figure 1. When a stream becomes available at t_0, we pick the next video among the videos with requests exceeding a pre-specified *batch threshold*, denoted by ν. (This eligible subset of videos is denoted by $L_{\mathcal{H}}$.) More specifically, we consider two stream selection strategies: The first strategy chooses the video with the maximum queue length to increase the batching effect and is thus called the *Max_Batch MQL* (or *BMQ*) scheme. Since the video with the maximal queue may not have the largest number of long waiting requests which are most likely to leave, the other strategy chooses the video that will incur the maximal expected loss if delayed until the next stream completion instant t_1 (i.e., the next time a stream becomes available). This is referred to as the *Max_Batch with minimal loss* (or *BML*) scheme. With no explicit knowledge of the viewer behavior in calculating the expected loss, the scheduler simply assumes that any viewer reaching the batch threshold will depart. (Therefore a video will become eligible for scheduling as soon as some viewer reaches the batch threshold.) Under this assumption, the expected number of departures (denoted by $l_k(t_1 - t_0)$) during the interval $[t_0, t_1)$ in the k-th video queue can be easily calculated based on the arrival time of each request in the queue. In our experimental studies, different viewer behaviors were considered, but the scheduler always used this simplified assumption. If at the stream completion time there is no viewer exceeding the batch threshold, the stream remains idle. The schedule is delayed until one of the viewers reaches the batch threshold.

The batch threshold parameter needs to be determined. While a natural choice is to assign to it the size of the wait threshold (i.e. $\nu = \omega$), our experimental studies (as described in the next section) showed that as the viewer wait threshold increases, choosing the maximal possible delay for batching (that does not incur loss of viewers) provides only a small increase in system throughput, and results in significant increase in the average response time. Thus, the Max_Batch schemes perform better by choosing $0 < \nu(\psi) \leq \omega$ for a desired *upper bound* ψ on the average response time. By optimizing on the value of ν we can optimize the Max_Batch schemes throughput while keeping the average response time small.

3.2 The Min_Idle schemes

The Min_Idle schemes are presented in Figure 2. Here the videos are partitioned in two sets. A video is considered to be in $\mathcal{H}$, if it satisfies one of the following criteria: (1) it is a hot video, (2) its queue has more than one request, and (3) the only request in its queue has wait time that exceeded the batch threshold (i.e. the difference between arrival time a_1 and scheduling time t_0 exceeds the batch threshold). Otherwise, a video belongs to the set $\mathcal{C}$. Videos in $\mathcal{H}$ will be delayed to achieve batching, but will be given also preferential treatment in scheduling when the batch threshold is approaching. Criterion (2) above provides a means to recognize hot videos dynamically, and reduces the sensitivity to the classification threshold τ. Criterion (3) brings the long waiting cold video requests into the $\mathcal{H}$ set to avoid cold video requests being completely blocked out by the hot video showings.

When a video stream becomes available we determine the eligible subset ($L_{\mathcal{H}}$) of videos which have requests with wait times that will exceed the batch threshold upon the next stream completion. If $L_{\mathcal{H}}$ is not empty, one of the videos in $L_{\mathcal{H}}$ will be chosen. The selection criterion can again be either the maximum queue length (referred to as the *IMQ* scheme) or the maximum expected loss (referred to as the *IML* scheme) as in the Max_Batch schemes. If $L_{\mathcal{H}}$ is empty, a cold video in the set $\mathcal{C}$ will be chosen. Since videos with more than one outstanding requests are in $\mathcal{H}$, each video in $\mathcal{C}$ can have a queue of length at most one. Hence the FCFS scheme is used.

The rationale for maintaining the sets $\mathcal{H}$ and $\mathcal{C}$ in the Min_Idle schemes is that different scheduling criteria can be applied to hot and cold videos. Hot videos (or any video with a queue length of two or more) will be scheduled only if further delay will result in viewers exceeding the batch threshold. This will improve batching. If no hot video viewer is going to exceed the batch threshold before the next stream completion, a cold video in $\mathcal{C}$ is scheduled. Since it is hard to achieve batching effect for cold videos, these video requests will be scheduled as soon as there is additional stream bandwidth available (when no hot video needs to be scheduled immediately). This is in contrast to the Max_Batch schemes, where scheduling of every video is delayed as much as possible. To avoid cold video requests being completely blocked out by the hot video showings, Min_Idle schemes also transfer the long waiting cold video requests into set $\mathcal{H}$.

Except for the BMQ scheme, all the other schemes (BML, IML and IMQ) take into consideration the next stream completion time in addition to the viewer wait tolerance in making the scheduling decision. Information on the next stream completion time can be used to help decide whether a viewer can be further postponed. We note that batching or any other stream sharing schemes generally work most effectively when VCR-like functions are not supported. In this case, the completion schedule can be easily calculated since the playout time of each video is known a priori.

When the VCR-like functions need to be supported, the completion schedule has to be dynamically maintained. Since we are only looking for the next stream completion time (within the range of the batch threshold), the prediction of completion time is made nearly at the end of a video showing, not at the beginning. Thus the probability of error and the effect of uncertainty are greatly reduced.

4 Simulation Study

4.1 Model description

We assume that video requests are generated by a Poisson process and the interarrival time is exponentially distributed with mean $1/\lambda$. Upon arrival to the system, a viewer chooses to wait for a showing of video k with probability p_k $, 1 \leq k \leq M$, with the p_k's given by the *Zipf* distribution: $p_k = \frac{c}{k}$, where $c = 1/(\sum_{k=1}^{M} \frac{1}{k})$ is a normalization constant. A viewer may leave the system without being served. Two scenarios of viewer reneging behavior are considered. In the first scenario, there is a minimum amount of time, ω, every viewer is willing to wait before reneging, while in the second one, no such time exists.

- **Scenario 1 (Deterministic case):** After waiting an amount of time given by the wait threshold ω, the viewer's remaining time until departure is given by the rv R, which is determined by an Exponential distribution. That is to say, $R \sim exp(\mu)$, where $1/\mu$ is the mean time that elapses after ω until the viewer leaves the system. Thus the viewer's overall wait time (W) is given by $W = \omega + R$.

- **Scenario 2 (Probabilistic case):** Viewer's departure time W follows a Normal distribution with mean μ and variance σ^2, i.e. $W \sim N(\mu, \sigma^2)$. A *confidence parameter* ζ is used by the scheduler for determining a wait threshold ω for the viewers. The value of ω is defined as

$$\omega = \min\{r \;:\; \mathrm{Prob}(W > r) \leq 1.0 - \zeta\} \quad , \tag{6}$$

i.e., the value of ω is chosen as the minimal amount of time after which a viewer is likely to leave the system immediately with probability higher than ζ.

Obviously, taking ζ close to 1 causes larger loss of viewers, while a small value for ζ decreases the batching effect in the system, since the viewers' wait tolerance allows further delay in their schedule.

4.2 Performance Results

We first describe the set of base parameter values used in the simulations. Assuming a system of $M = 100$ videos with the average length of a showing $L = 120$ minutes,[2] we used a server capacity of 800 streams and a video request rate of 50 arrivals per minute.

The classification threshold for distinguishing between videos in $\mathcal{C}$ and in $\mathcal{H}$ was chosen to be $\tau = 20$. For Scenario 1, the wait threshold is assumed to be $\omega = 10.0$ minutes and the average remaining time till departure was 6 minutes (that is, $R \sim exp(1/6)$). We set the batch threshold equal to the wait threshold, i.e. $\nu = \omega$ (unless otherwise specified). For Scenario 2, with the reneging pattern that corresponds to the Normal distribution, we use the value $\mu = 10.0$ minutes as mean for W with the variance $\sigma^2 = 2.5$ minutes.

In Figures 3–11 we compare the performance of the proposed schemes to the MQL and FCFS schemes. In these experiments the four proposed schemes (IML, IMQ, BML and BMQ) performed much closer to each others than MQL and FCFS, thus we plotted only the numerical results obtained for IML. This makes the graph more readable and provides a preliminary distinction between the previously studied

[2]To simplify the performance study, we do not assume any VCR-like functions being supported.

schemes and our wait tolerance batching approach. Figures 12–14 demonstrate some of the differences in performance among the four proposed schemes.

In Figure 3 we show the turn-away probability vs. arrival rate to the VOD system. The decrease in turn-away probability under IML is noticeable even for low arrival rates. As the load increases, the turn-away probability under FCFS approached 12% while under IML this probability was at most 6%. Figure 4 shows that IML also provides smaller average response time compared to FCFS. We note that MQL yields a typical response time of at most 1.5 minutes at the cost of causing more than 25% of the viewers to renege before being served (see Figure 3). In Figure 5 we examine the fairness issue. The unfairness of IML is close to that of FCFS and is much better than the unfairness that was measured for MQL.

An alternative way for quantifying the improvement achieved by the wait tolerance batching approach in terms of stream utilization is to measure the minimal server capacity required for keeping the turn-away probability in a desired range. In Figure 6 we compare the minimal server capacity, that guarantees a turn-away of at most 5% of the viewers, under FCFS and IML. For increasing arrival rates, IML reduces the required server capacity in almost 15%. We comment that the improvement over MQL is much more significant as can be deduced from Figure 3. (In fact, the curve we obtained was out of the range of Figure 6.)

In Figure 7 we test the sensitivity of the proposed schemes to changes in the classification threshold τ. While our initial choice of τ for IMQ and IML may seem somewhat arbitrary, Figure 7 shows that the Min_Idle schemes are insensitive to that choice. We plotted the throughput achieved by the two schemes, where $\tau = 5, \ldots, 90$, otherwise setting the system parameters to their base values. The corresponding throughput varied in the small range of 48.7–49.9. This is due to the fact that both IMQ and IML use additional criteria (that depend on the states of the wait queues) for defining *dynamically* the partition of videos to $\mathcal{H}$ and $\mathcal{C}$.

In Figures 8 and 9 we show the effect of changes in the viewer wait threshold on turn-away probabilities and average response times, respectively, for IML, FCFS and MQL. Here we tune the batch threshold to maximize the system throughput while maintaining a small response time. For wait thresholds in the range $[0, 20]$ minutes and optimal choices of corresponding batch thresholds (see Figure 10), IML decreases the turn-away probability in up to 30% compared to FCFS while keeping the average response time bounded by 6.5 minutes. We note that IML achieves substantial improvement over FCFS at small wait thresholds, while for a large wait threshold (say, 15 minutes), the turn-away probabilities under both IML and FCFS approach zero and the difference in response times diminishes. Figure 10 shows the optimal batch threshold vs wait threshold. As wait threshold ω increases, the optimal choice of batch threshold becomes smaller than the wait threshold to satisfy the response time constraint. We note that choosing the maximal possible delay (ω) for batching provides only a small increase in system throughput, and results in a significant increase in the average response time. This is why an additional parameter, batch threshold ν, is introduced.

Next we use Scenario 2 to study the effect of uncertainty on the wait tolerance. We consider the case where $W \sim N(10.0, 2.5)$. The value of the confidence parameter (ζ) used by the scheduler is 0.75. Figure 11 shows the turn-away probabilities of IML, MQL and FCFS under different server capacities. We conclude that even with uncertainty on the wait threshold, IML outperforms MQL and FCFS. (In fact further experiments showed that when varying the the confidence parameter from 0.35 to 1, the changes on turn-away probability are within 5%.) A decrease of more than 40% of turn-away probability can be achieved by IML for server capacities of 500 streams or less.

In Figures 12 and 13 we present some distinctions between the Max_Batch schemes and the Min_Idle schemes. Figure 12 shows the advantage of the Min_Idle schemes in providing inherently smaller average response time for low arrival rates. We note that for Max_Batch schemes, the response time is worst for small loads:

every request needs to wait out the batch threshold. When the load increases, the response time somewhat improves, as an arriving viewer is likely to find his queue non-empty, and therefore will be scheduled before exhausting the batch threshold. At higher loads, both schemes converge to similar response time. Min_Idle has lower response time at no load because the cold video requests do not need to wait. As the load increases, its response time slightly improves, and then goes up again due to stream contention. Figure 13 indicates that for a wait threshold of 10.0 the Min_Idle schemes also require less server capacity for guaranteeing a turn-away probability of at most 5%.

Figure 14 provides a distinction between the MQL-based criterion (BMQ and IMQ) and the Minimal-Loss-based criterion (BML and IML) for stream selection. We note that as the load increases, the MQL-based schemes yield higher unfairness compared to the Minimal-Loss based schemes. Again this shows a side effect on over-emphasis on batching. Furthermore, BML (respectively, BMQ) shows slightly smaller unfairness than IML (respectively, IMQ) although at the expense of a bit higher minimal server capacity requirement as shown in Figure 13. This is due to the fact that by imposing a minimum wait time requirement, the Max_Batch schemes in fact limit the number of streams that can be simultaneously used by each video. (The upper limit is set by the length of a video divided by the batch threshold.) This prevents the hot videos from using up all the streams and improves fairness. The Min_Idle schemes do not impose such a wait time restriction but allow for dynamic adjustment of the cold videos into set $\mathcal{H}$.

5 Conclusion

We have presented a paradigm for multimedia scheduling schemes that explore viewers wait tolerance for increasing the system throughput. Based on this approach we have introduced four schemes, which provide substantial improvement compared to previously studied FCFS and MQL schemes, especially for scenarios of heavy load for which the efficient use of server capacity is crucial in providing good quality of service.

The four alternatives schemes (IML, IMQ, BML and BMQ) proposed differ not only in their eligibility criteria for scheduling but also in stream selection rules. The eligibility criterion is introduced to facilitate batching. We found that an aggressive pursue of batching may not lead to the best performance. Other factors like response time, expected viewer losses and stream idling also need to be taken into consideration.

Under the Max_Batch (BML and BMQ) schemes, videos are not eligible for scheduling until they have viewers with wait time exceeding the batch threshold. The Min_Idle (IML and IMQ) schemes relax the eligibility criterion: Batching is only pursued for the hot videos without imposing the requirement to wait out the whole batch threshold. Cold videos are always eligible for scheduling under the Min_Idle schemes. These schemes provide inherently smaller average response time when the request rate is low (i.e. non-peak hours) and require less server capacity for guaranteeing a given turn-away probability.

Two streams selection criteria were considered. We studied the trade-offs between using MQL criterion (BMQ and IMQ) and Minimal-Loss criterion (BML and IML) for stream selection. It is found that as the load increases, the MQL based schemes that try to optimize batching yield higher unfairness compared to the Minimal-Loss based schemes.

References

[1] E. F. Beckenbach, Editor, Applied Combinatorial Mathematics, J. Wiley and sons, 1964.

[2] A. Dan, D. Sitaram and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Sever with Batching", *Proc. ACM Multimedia '94*, SF, CA, Oct. 1994, pp. 391-398.

[3] L. Golubchik, J.C.S. Lui and R. Muntz, "Reducing I/O Demand in Video-on-Demand Storage Servers", *Proc. ACM SIGMETRICS '95*, Ottawa, Ontario, Canada, May 1995.

[4] M. Kamath, D. Towsley and K. Ramamritham, "Buffer Management for Continuous Media Sharing in Multimedia Database Systems", *Technical Report 94-11*, Univ. of Massachusetts, Feb. 1994.

[5] P. V. Rangan and H. M. Vin, "Designing File Systems for Digital Video and Audio", *Proc. of 12th ACM Symposium on Operating Systems*, 1991.

[6] W. Sincoskie, "System Architecture for Large Scale Video on Demand", *Computer Networks ISDN System*, Vol. 22, 1991, pp. 155-162.

[7] D. Rotem, and J.L. Zhao, "Buffer Management for Video Database Systems", *Proc. Intl. Conf. on Data Eng.*, Taipei, Taiwan, March 1995, pp. 439-448.

[8] H. M. Vin and P. V. Rangan, "Designing a Multi-User HDTV Storage Server", *UCSD Technical Report CS92–225*, Jan. 1992.

[9] P. S. Yu and J.L. Wolf and H. Shachnai, "Look-ahead Scheduling to Support Pause-Resume for Video-on-Demand Applications", *Proc. Multimedia Computing and Networking 1995, (IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology)*, San Jose, CA, Feb. 1995.

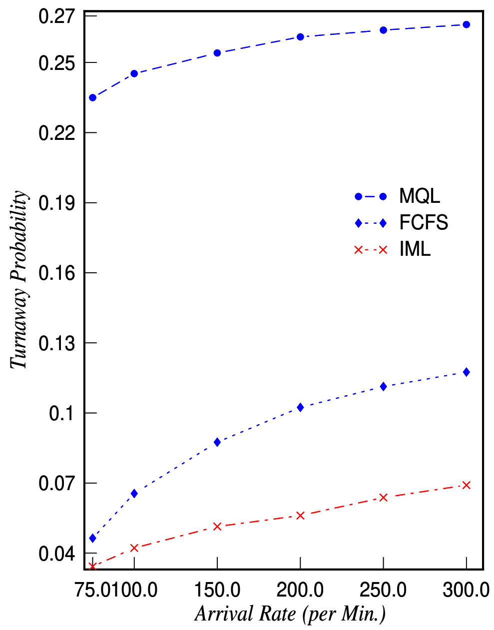

Figure 3: Turn-away Probability vs. Arrival Rate (Comparison to FCFS and MQL)

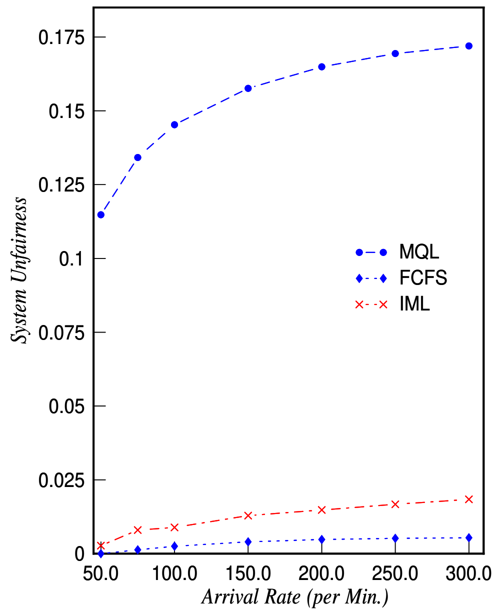

Figure 5: Unfairness vs. Arrival Rate

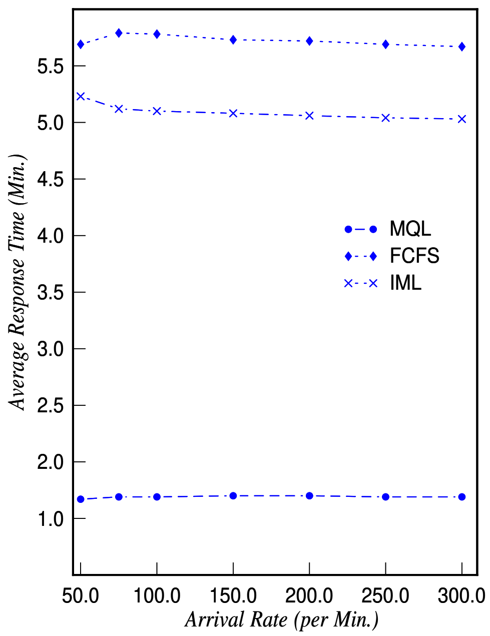

Figure 4: Average Response Time vs. Arrival Rate

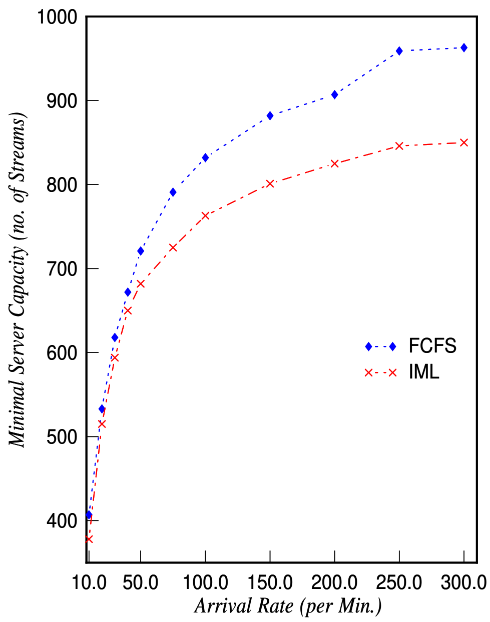

Figure 6: Minimal Server Capacity For a Turn-away of less than 5%

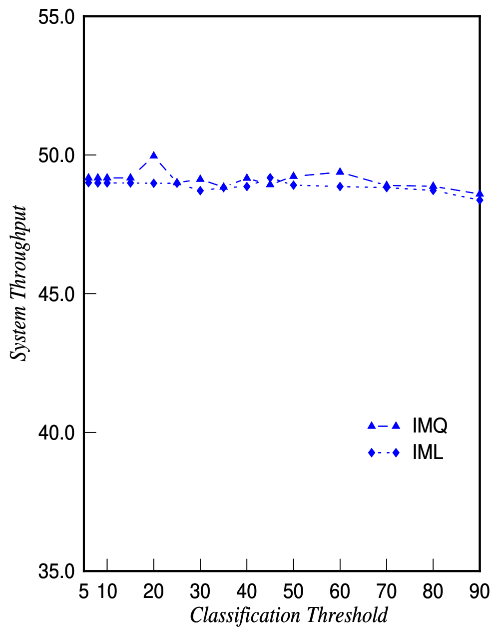

Figure 7: Throughput of the IMQ and IML Schemes with Changing Classification Threshold

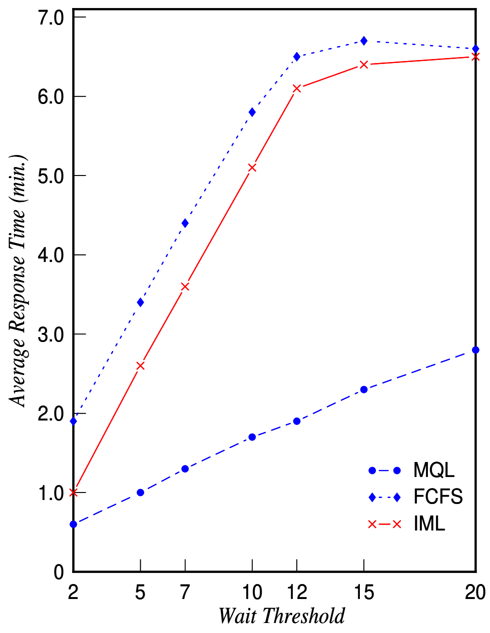

Figure 9: Average Response Time with Varying Wait Thresholds and Optimal Batch Thresholds (min.s)

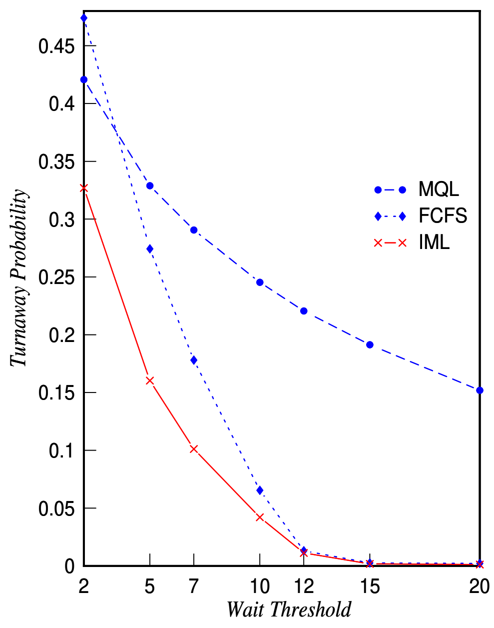

Figure 8: Turn-away Probability with with Varying Wait Threshold and Optimal Batch Thresholds (min.s)

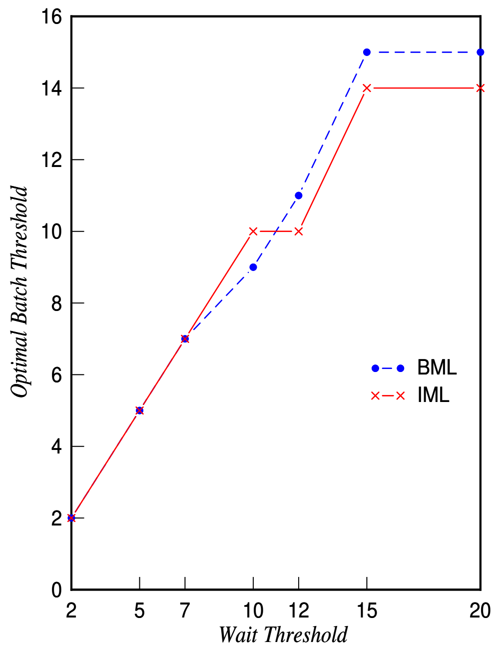

Figure 10: Optimal Batch Thresholds for Varying Wait Thresholds (min.s)

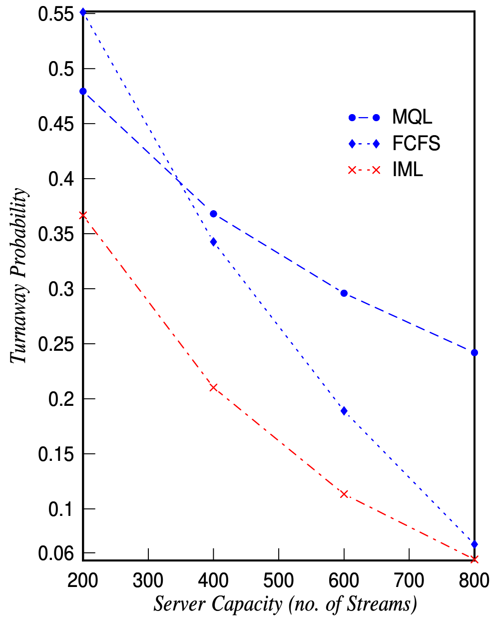

Figure 11: Turn-away Probability vs. Server Capacity (Reneging by the Normal Distribution)

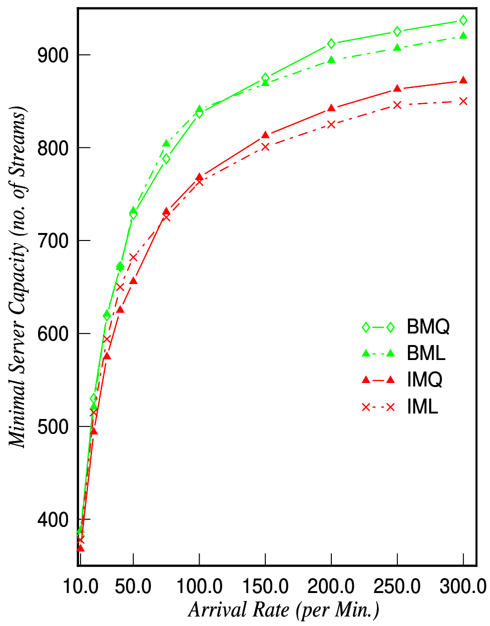

Figure 13: Minimal Server Capacity For a Turn-away of less than 5% (the Max_Batch Schemes vs. The Min_Idle Schemes)

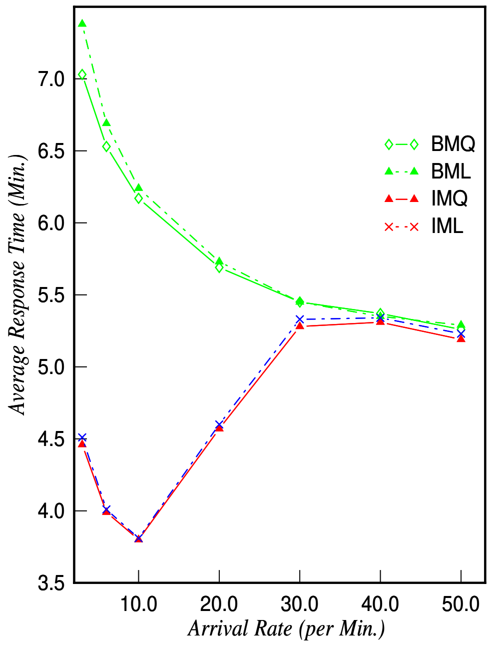

Figure 12: Average Response Time for Varying Arrival Rate (the Max_Batch Schemes vs. the Min_Idle Schemes)

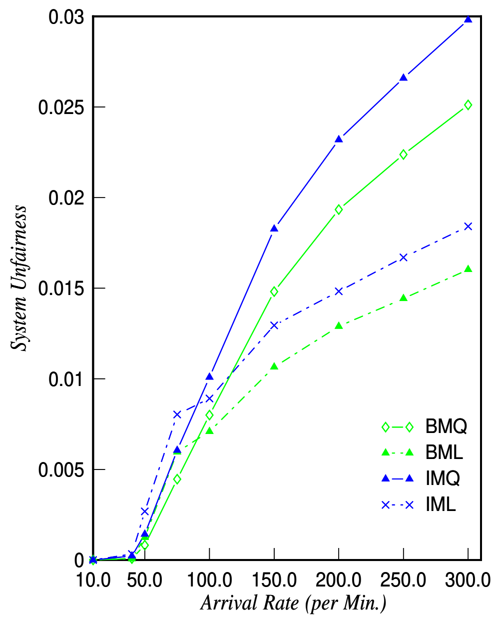

Figure 14: System Unfairness for Varying Arrival Rate (the Max_Batch Schemes vs. The Min_Idle Schemes)