Group-Guaranteed Channel Capacity in Multimedia Storage Servers

Athanassios K. Tsiolis
tsiolis@cs.wisc.edu

Mary K. Vernon
vernon@cs.wisc.edu

Computer Sciences Department
University of Wisconsin-Madison
1210 West Dayton Street
Madison, WI 53706, U.S.A.

Abstract

One of the open questions in the design of multimedia storage servers is in what order to serve incoming requests. Given the capability provided by the disk layout and scheduling algorithms to serve multiple streams simultaneously, improved request scheduling algorithms can reduce customer waiting times. This results in better service and/or lower customer loss. In this paper we define a new class of request scheduling algorithms, called Group-Guaranteed Server Capacity (GGSC), that preassign server channel capacity to groups of objects. We also define a particular formal method for computing the assigned capacities to achieve a given performance objective. We observe that the FCFS policy can provide the precise time of service to incoming customer requests. Under this assumption, we compare the performance of one of the new GGSC algorithms, $GGSC_W$-FCFS, against FCFS and against two other recently proposed scheduling algorithms: Maximum Factored Queue length (MFQ), and the FCFS-n algorithm that preassigns capacity only to each of the *n* most popular objects. The algorithms are compared for both *competitive market* and *captured audience* environments.

Key findings of the algorithm comparisons are that: (1) FCFS-n has no advantage over FCFS if FCFS gives time of service guarantees to arriving customers, (2) FCFS and $GGSC_W$-FCFS are superior to MFQ for both competitive and captive audience environments, (3) for competitive servers that are configured for customer loss less than 10%, FCFS is superior to all other algorithms examined in this paper, and (4) for captive audience environments that have objects with variable playback length, $GGSC_W$-FCFS is the most promising of the policies considered in this paper. The conclusions for FCFS-n and MFQ differ from previous work because we focus on competitive environments with customer loss under 10%, we assume FCFS can provide time of service guarantees to all arriving customers, and we consider the distribution of customer waiting time as well as the average waiting time.

1 Introduction

Recent advances in computer system and networking technologies have made the design and implementation of commercial multimedia servers feasible. A few prototypes exist [19], and many more are on the drawing boards.

The multimedia server application of interest to this paper is video-on-demand, where customers request playback of particular objects available on the server. Possible environments vary from highly competitive large-scale enterprises that are aimed at capturing a reasonable share of the home market, to smaller-scale captive-audience enterprises such as a campus multimedia server that contains movies and course-related multimedia objects, or corporate servers that contain multimedia clips for use in various technical and marketing activities of a given firm. The Time Warner system recently introduced in Orlando [18], is an example of the envisioned large-scale competitive environments.

The resources that are typically required for a multimedia server are shown in figure 1. Large storage requirements (e.g., 4 gigabytes of storage for a single MPEG-2 encoded two-hour movie), variability in the customer demands, and possibly real-time delivery constraints (e.g., typically 4.5-800 Mbps continuous delivery rate per segment, depending on the required image quality), add up to challenging issues for the design of cost-effective servers. Vendors are seeking solutions. Careful performance evaluations of the design options can contribute to the effort.

Several papers in the literature address the challenging issues of data layout, disk scheduling [4, 14, 13, 9, 3, 6] and buffer management [8, 17, 16, 10]. Intelligent data layout and disk scheduling, such as the staggered striping algorithm proposed by Berson et. al. [4], increases the number of multimedia streams that can be delivered simultaneously from a given disk or collection of disks. Sophisticated buffer management algorithms, such as the one proposed by Dan and Sitaram in [8], can further increase effective disk bandwidth by serving some streams from available primary memory.

Assuming that customer request arrivals are bursty, and thus there are (short) periods during which demand exceeds server capacity, another challenging resource management issue for multimedia servers is in what order to serve incoming requests. Given the capability provided by the disk layout and scheduling to serve multiple streams simultaneously, improved request scheduling can reduce customer waiting times. This in turn results in better service and/or lower customer loss.

Several previous papers have investigated request scheduling policies for multimedia servers [11, 15, 7, 5, 2, 1]. One difficulty in evaluating the algorithms is that the characteristics of multimedia server requests are currently unknown. Only a few prototype systems are in existence. Workloads on these systems have not been reported and may not be representative of future workloads anyway. Nevertheless, the previous work has shown that proposing and comparing various possible request scheduling algorithms under plausible workloads can provide designers of early systems with some useful guidance. For example, if the simple FCFS scheduling policy does not provide time of service guarantees to incoming re-

This research was supported in part by grants from the National Science Foundation (EHR-95-50429, CCR-9024144).

TO APPEAR IN THE SIGMETRICS 1997 CONFERENCE PROCEEDINGS.

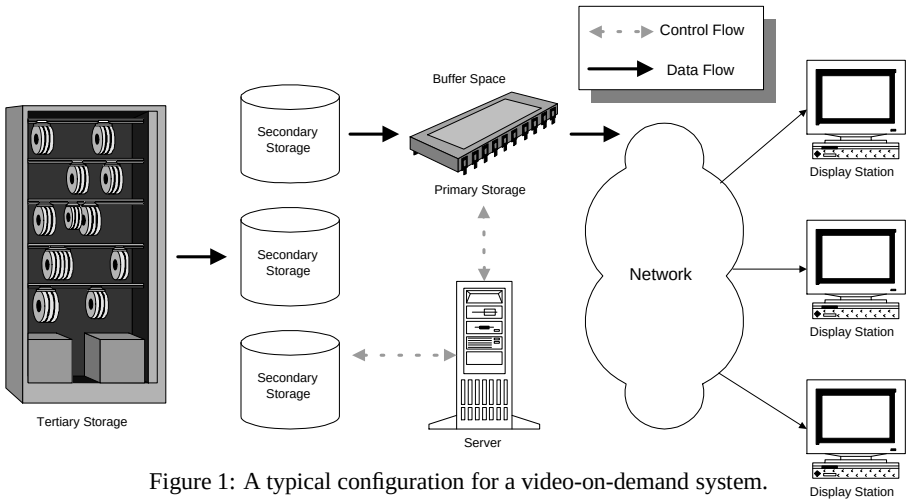


Figure 1: A typical configuration for a video-on-demand system.

quests, Dan, Sitaram, and Shahabuddin have shown that reserving some of the available channels to guarantee good service to the *n* most popular objects (FCFS-n) can lead to significant gains in performance in the case that time of service guarantees can influence customers to wait longer for service [11]. More importantly, a recent paper by Aggarwal, Wolf, and Yu [1] shows that an algorithm called Maximum Factored Queue length (MFQ) minimizes average wait at full capacity, and also has lower customer loss than FCFS for particular points in the system design space. Another benefit of the early investigation of effective request scheduling policies is that these policies may suggest improved data layout, disk scheduling, and/or buffering algorithms. We comment on this further in section 6.

This paper further investigates request scheduling for video-on-demand servers. In particular, we

- propose a new family of algorithms (Group-Guaranteed Server Capacity, GGSC) in which server capacity is preassigned to groups of objects, with a method of computing the assigned capacity per group to meet a given performance objective,
- observe that the FCFS scheduling algorithm can provide a time of service guarantee to an incoming request,
- compare a particular GGSC policy, $GGSC_W$-FCFS, against the previously proposed MFQ, FCFS, and FCFS-n policies, under the new assumption that FCFS can provide time of service guarantees to arriving customers, and
- evaluate the algorithms for (1) *competitive market* environments where the key performance metrics are (a) system cost vs. target customer loss (e.g., less than 1%) and (b) customer loss vs. object popularity; and (2) *captured audience* environments, where the key performance metrics are mean and *distribution* of waiting time, overall and as a function of object popularity.

Key findings of the algorithm comparisons are that: (1) FCFS-n has no advantage over FCFS if FCFS gives time of service guarantees to arriving customers, (2) FCFS and $GGSC_W$-FCFS are superior to MFQ for both competitive and captive audience environments, (3) for competitive servers that are configured for customer loss less than 10%, FCFS is superior to all other algorithms examined in this paper, and (4) for captive audience environments that have objects with variable playback length, $GGSC_W$-FCFS is the most promising of the policies considered in this paper.

The remainder of this paper is organized as follows. Section 2 defines the previously proposed and new request scheduling policies that are studied in this paper. A particular approach for assigning channel capacities in the GGSC algorithm is given in section 3. Section 4 discusses the competitive market and captured audience environments that multimedia servers may operate in. The assumptions about customer behavior and metrics that will be used to evaluate the policies for those environments are also discussed in section 4. Section 5 contains the results of the policy comparisons (using simulation), and section 6 provides the conclusions and future directions for this work.

2 Request Scheduling Algorithms

Below we review the first-come first-served (FCFS) algorithm, a related family of scheduling algorithms known as FCFS-n which were proposed in [11], and the maximum factored queue length first (MFQ) policy described in [1]. We then define a new family of scheduling algorithms, *Group Guaranteed Server Capacity (GGSC)*. These algorithms will be compared in this paper. Another previously proposed algorithm, maximum queue length first (MQL), is not considered in this paper since previous work has shown that MQL has some undesirable performance characteristics [11], and MFQ is more intelligently aggressive in scheduling the more popular objects for playback [1].

In defining each algorithm, we will use the term *channel* to denote the set of resources needed to deliver a multimedia stream. As discussed in section 1, we assume that multiple channels can operate simultaneously. We also assume that the network delivering the stream to the customer has a multicast facility, so that one channel can deliver a stream to multiple clients. Dan et. al. have shown that extra *contingency channels* can be used effectively to handle pausing and fast-forwarding by clients who are viewing a multicast object [7]. The algorithms evaluated in this paper deal only with scheduling the primary channels and assume that multiple requests for an object can be scheduled on a single channel.

In each algorithm, the times at which channels become available are *persistently staggered* such that these times are uniformly distributed over an interval equal to the fixed or average playback time of the objects that the set of channels can serve. Such staggering enforces more uniform quality of service over time. Staggering channel availability also promotes batching of requests for more efficient use of the channels.

2.1 FCFS and FCFS-n

In the FCFS scheduling algorithm, requests are served in first-come first-served order as channels become available. When a request is served, all other clients waiting for the same object are also served.

In the FCFS-n family of algorithms, a set of channels are pre-allocated to each of the n objects that are requested most frequently. The objects with pre-allocated capacity are called the *hot* objects. For each hot object, enough channels are allocated so that a channel becomes available to play the object every T units of time. T and n are parameters of the policy. A request for a hot object is served only by the channels reserved for it. However, when a reserved channel becomes available and there are no clients waiting for the hot object to play, the reserved channel can serve a request for a cold object. Requests for cold objects are served in FCFS order by the unreserved channels and by reserved channels that would otherwise go unused. The advantages of the reserved server capacity are: (1) requests for a popular object are batched to make more effective use of the channel, and (2) requests for hot objects are given a time of service guarantee that, if small enough, may influence the client to wait for the request to be served. The objective is to set T equal to some reasonable value that customers requesting the object will be willing to wait. Previous work has investigated T equal to 2 or 5 minutes.

We observe that the simple FCFS scheduling algorithm can also provide a time of service guarantee to arriving requests. The actual time of service could be sooner than the given guarantee in two cases. First, if other customers already in the system decide to leave before their request is served. Second, in FCFS-n, service for cold objects can occur sooner than the time of service guarantee if one of the channels reserved for a hot object becomes available to serve a cold object. Otherwise, the time of service guarantees in FCFS are precise, and if the guaranteed time of service is small enough the customer may be influenced to wait for the request to be served. The observation that FCFS can provide time of service guarantees to all customers was not taken into account in previous comparisons of the FCFS and FCFS-n algorithms; those comparisons showed FCFS-n to be superior primarily in cases where its time of service guarantees influenced customer behavior [11].

2.2 Maximum Factored Queue Length (MFQ)

The objective of MFQ is to optimally batch customer requests by scheduling the object with the maximum *factored queue length*, where the factored queue length is a weighted queue length designed to minimize overall mean waiting time for customer requests [1]. This factored queue length is derived as follows for a system in which all objects have the same playback length, L.

If T_m is the average time between two successive playbacks of object m and the customer arrivals are a Poisson process, the average waiting time for a customer of object m is $T_m/2$. In this case, we want to minimize the quantity:

$$\sum_{m=1}^{M} f_m \cdot T_m/2,$$

where f_m is the fraction of requests that are for object m.

Let C denote the total number of available channels. At full capacity, the number of channels per unit time scheduled by the server is C/L, and the average number of channels allocated to object m per unit of time is $1/T_m$. The factored queue length for MFQ is derived by solving the following optimization problem:

Minimize

$$\sum_{m=1}^{M} f_m \cdot T_m$$

$$\text{such that} \quad \sum_{m=1}^{M} 1/T_m = C/L$$

$$\text{and} \quad T_m > 0.$$

As explained in [1], the optimal solution occurs when:

$$T_1\sqrt{f_1} = T_2\sqrt{f_2} = \ldots = T_M\sqrt{f_M}. \tag{1}$$

The average batch size for object m, B_m, is equal to $f_m \lambda T_m$ if customers never defect before receiving service. Substituting in (1) we have:

$$\frac{B_1}{\sqrt{f_1}} = \frac{B_2}{\sqrt{f_2}} = \ldots = \frac{B_M}{\sqrt{f_M}}. \tag{2}$$

MFQ greedily approximates the above condition for minimum average waiting time in a system that operates at capacity and with no customer loss, by assigning the next available channel to the object with the maximum $q_m/\sqrt{f_m}$, where q_m is the number of customers waiting for object m when the channel becomes available.

If the object request frequency, f_m, is not known its instantaneous value can be approximated by $q_m/\Delta T_m$, where ΔT_m is the time since object m was last scheduled for playback. In this case, the object to be scheduled when a channel becomes available is the one with the maximum: $q_m \cdot \Delta T_m$. We call this policy MFQ-instant-f.

Note that [1] states that the above derivation holds for systems with unequal object lengths, where L above denotes the average playback length. Thus, they claim, the same factored queue lengths will minimize mean wait for unequal length objects. However, this is incorrect. In section 3.2 we show how to derive the correct factored queue lengths for unequal object playback times.

2.3 Group-Guaranteed Server Capacity (GGSC)

The MFQ algorithm has the nice property of optimal batching to achieve minimum average waiting time for object playback when the system operates at capacity and there are no defections. However, MFQ cannot give time of service guarantees. Such guarantees make waiting easier and may induce a customer to wait for service who would otherwise grow impatient and leave. The Group-Guaranteed Server Capacity (GGSC) family of algorithms was motivated by the desire to optimize batching yet also give time of service guarantees.

Near-optimal batching is achieved in GGSC by preassigning server capacity to each object in order to achieve an objective function, such as minimizing mean wait. Time of service guarantees are achieved by grouping together objects that have nearly equal expected batch size for the given capacity assignments and request arrival rate, and then scheduling requests in each group in FCFS order on the collective channels that are assigned to the group. In GGSC_W-FCFS, the GGSC algorithm that minimizes mean wait at full capacity, expected batch size decreases as object popularity decreases. Objects are grouped, starting from the most popular object, such that the difference in expected batch size between the most popular object in the group and the least popular object is less than or equal to one. Section 3.1 describes how channel capacity is assigned to minimize mean wait and also gives an example of the resultant channel assignments, expected batch sizes, and object grouping.

To simplify channel staggering, only objects of the same or similar playback length are grouped together in GGSC algorithms. Once channel capacity is assigned to a group of objects, the times

at which the channels become available are persistently staggered over the period equal to the playback length for objects in the group. If no requests are waiting for any object in a group when one of its preassigned channels becomes available, the channel will serve a waiting request from another group as long as its playback length is less than the time until the channel must be made available again to its designated group.

The key parameters of the GGSC algorithm are the performance objective that is used to compute assigned capacities, and the algorithm for deciding which request(s) outside the group to serve when there are no requests waiting for objects in the group. A key challenge is to determine which performance objective is most effective and to distribute the capacity to meet the objective. In the next section, we explain how server capacity is assigned in order to minimize the average waiting time. The GGSC algorithm with this objective function, and with FCFS scheduling of requests from alternate groups, will be evaluated in this paper. We call this algorithm $GGSC_W$-FCFS.

3 Optimal Channel Capacity Assignments

3.1 $GGSC_W$-FCFS Channel Assignments

In this section we outline how channels are preassigned to each object in order to minimize the average waiting time for requests when the system is operating at full capacity (i.e., when requests are scheduled only on the channels assigned to their group).

If C_m is the number of channels designated to object m and L_m is the playback duration of object m, then the dedicated channels will become available to play the object every $T_m = L_m/C_m$ units of time. We assume that the fraction of customers that request object m, f_m, is known or can be measured for each of the M objects on the server. Channel capacity is allocated to minimize the average client waiting time by solving the following optimization problem:

Minimize

$$\sum_{m=1}^{M} f_m \frac{L_m}{C_m} \tag{3}$$

$$\text{subject to} \quad \sum_{m=1}^{M} C_m \leq C, \tag{4}$$

where C is the total number of channels. Note that to minimize the overall mean wait all of the channels will be distributed among the available objects.

The above optimization problem can be solved analytically, using the Lagrange multiplier approach. Define:

$$\mathcal{S} = \sum_{m=1}^{M} f_m \frac{L_m}{C_m} - \gamma \left(C - \sum_{m=1}^{M} C_m \right) \tag{5}$$

where γ is the Lagrange multiplier. The solution for the optimal channel assignments, $\{C_m\}$, must satisfy the following equations:

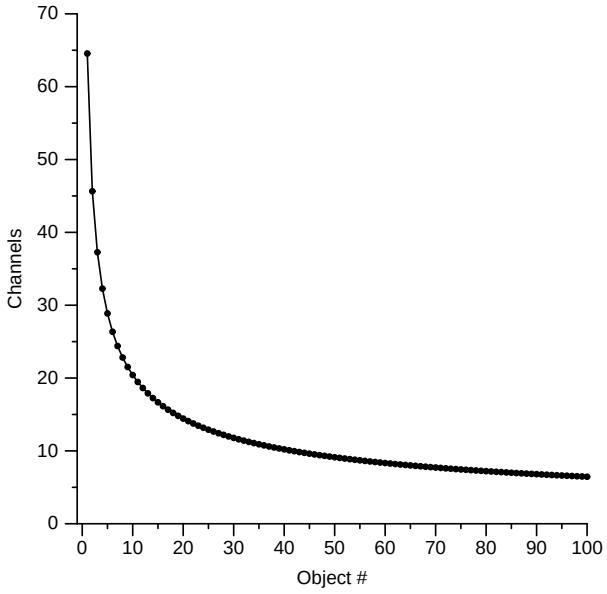
$$\frac{\partial \mathcal{S}}{\partial C_m} = -\frac{f_m L_m}{C_m^2} + \gamma = 0, \quad m = 1, \ldots, M \tag{6}$$

and

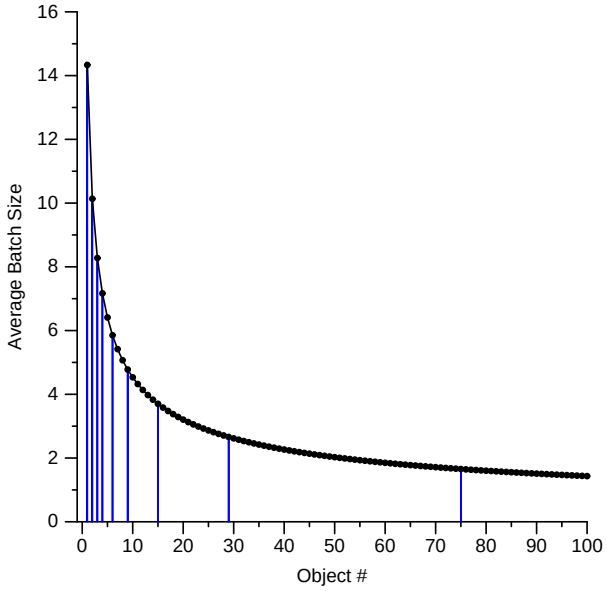
$$\frac{\partial \mathcal{S}}{\partial \gamma} = -\left(C - \sum_{m=1}^{M} C_m \right) = 0. \tag{7}$$

Solving equations (6) and (7) we get the optimal values for the C_m's:

$$C_m = \frac{\sqrt{f_m L_m}}{\sum_{i=1}^{M} \sqrt{f_i L_i}} \cdot C, \quad m = 1, \ldots, M \tag{8}$$



(a) Channel assignments (1200 channels)



(b) Expected batch size (40 customers/min)

Figure 2: Example channel assignments and batch sizes. (100 2-hour objects; Zipf(0) object selection frequencies)

The above channel assignments reserve more server capacity as object popularity increases. This means that, for a given object playback time, the expected waiting time will be lower for the more popular objects, which is to be expected given the objective function.

Example channel assignments are shown in figure 2(a) for the case of 1200 channels, one hundred objects of equal playback duration, and a Zipf(0) distribution of object selection frequencies. Figure 2(b) shows the expected batch size and channel groups for the same system and a request arrival rate of 40 customers per minute.

3.2 Optimal Batching in MFQ

We can derive the optimal factored queue lengths for MFQ when objects have varying playback lengths from the optimal channel assignments for $GGSC_W$ given in equation (8) above.

If T_m is the average time between successive playbacks of object m, from (8) we have:

$$T_m = \frac{L_m}{C_m} = \sqrt{\frac{L_m}{f_m}} \cdot \frac{\sum_{i=1}^{M} \sqrt{f_i L_i}}{C} \tag{9}$$

Thus:

$$\frac{T_m}{T_n} = \frac{\sqrt{L_m / f_m}}{\sqrt{L_n / f_n}} \tag{10}$$

or, given that $B_m = f_m \lambda T_m$,:

$$\frac{B_m}{\sqrt{f_m L_m}} = \frac{B_n}{\sqrt{f_n L_n}}, \quad \text{for} \quad 1 \le m, n \le M. \tag{11}$$

Equation 11, which reduces to equation (2) if all objects have the same playback duration ($L_m = L$), should be used to derive the factored queue lengths for MFQ when playback durations are unequal.

4 Server Environments and Workload Assumptions

The key questions to be addressed in this paper are:

- How does the performance of FCFS compare to that of the FCFS-n algorithms, given that both can provide tight waiting time guarantees to each arriving request?
- How does the performance of $GGSC_W$-FCFS compare to the performance of FCFS, FCFS-n, and MFQ?
- How do various assumptions about the server request workload affect the relative performance of the algorithms?

Below we describe the server environments, metrics, and workloads that will be considered in the policy evaluations in section 5.

4.1 Server Environments and Performance Metrics

We examine the above issues for two types of server environments. In the *competitive market environment* customers have a choice of service provider. If service is poor, a customer will shop around for a better server. Short waiting time is important to the customer. Minimizing customer loss is of paramount importance to the service provider. Such an environment might be typical for a large enterprise that wishes to capture a good share of a significant home or business market. Economy of scale allows for investment in high-end servers, but providers are interested in the most cost-effective way to provide the requisite service.

The *captive audience* environment includes organizations such as universities or individual companies that use multimedia servers for various purposes and have a limited budget. The audience will tolerate longer waiting times compared to customers in a competitive environment due to the lack of alternative servers. Prearranged times at which certain objects can be viewed may be very practical, if not desirable, in this environment. The principal goal in designing the server is the best possible quality of service that a given moderate cost can achieve.

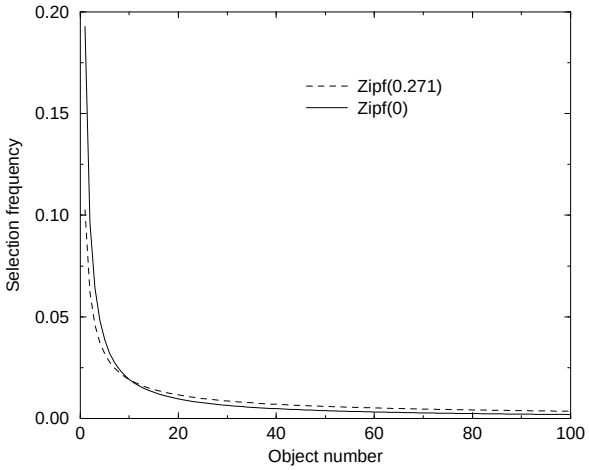


Figure 3: Zipf distributions of object selection frequency.

A variety of metrics will be used to evaluate the algorithms in each environment, but the principal metric for competitive environments will be the system cost (i.e., number of channels) required to achieve a given low target customer loss probability. In this case, the model of customer behavior must include a specification of how long the customer is willing to wait for service. If service cannot be delivered within that time, the customer cancels the request and searches for another provider.

The principal metrics for the captive audience environment are the mean and distribution of waiting time as a function of the number of channels and object popularity. We will find that these metrics will yield different conclusions about the relative performance and fairness of MFQ and FCFS than the conclusions found in previous work. In this environment, it will be assumed that customers always wait to be served.

4.2 Workload Assumptions

Specific workload assumptions that need to be specified include: the number of objects that are requested, the object playback durations and relative selection frequencies, the customer arrival rate, and the time that a customer is willing to wait for service in a competitive environment. As mentioned in section 1, workload data is not yet available since multimedia servers are still in the prototype stage of development. We thus make the assumptions outlined below, with the goal of obtaining some useful early intuition about the relative strengths and weaknesses of the policies, rather than obtaining precise quantitative performance estimates for an empirical set of workload parameters.

A multimedia server may have hundreds or thousands of objects that can be requested. Typically a large fraction of the objects will be kept on tertiary storage and such objects will have extremely low selection frequency. To gain an understanding of relative request scheduling policy performance we model only the objects that have selection frequencies above a given (small) selection frequency threshold.

Previous work [11] has shown that measurements of a large video movie outlet [12] yielded selection frequencies that were closely fit by the discrete Zipf(0.271) distribution illustrated in figure 3. In this distribution 100 objects have selection frequencies above 0.0035. In the absence of better data, and since the Zipf distribution has intuitive appeal for both the competitive and captive audience environments discussed above, we will use the Zipf distribution of selection frequencies, normalized for 100 objects. In most experiments we will use the Zipf(0) distribution, also illustrated in

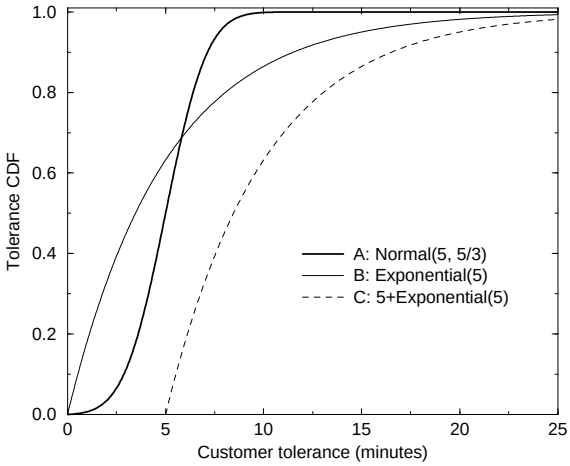


Figure 4: Customer tolerance distributions.

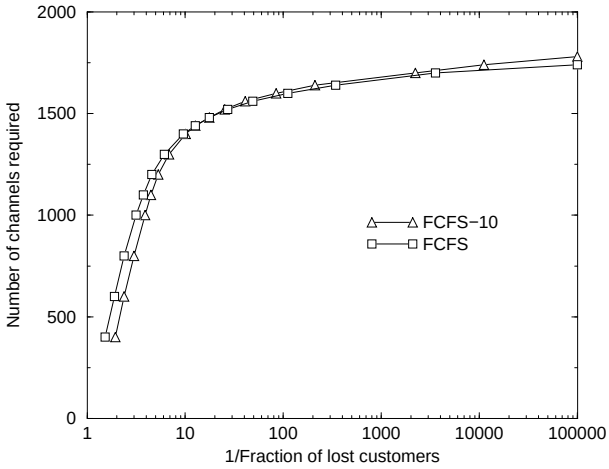


Figure 5: Performance of FCFS and FCFS-n.
(competitive environment; tolerance: 5 or model B)

the figure. One rationale for this higher-skew Zipf distribution is that a multimedia server can supply the most popular objects to all customers who request them, whereas customers at the video store sometimes make another selection when the object they want is not available. In a few experiments we use the Zipf(0.271) distribution in order to compare our results with previously published results.

The population of customers for a multimedia server is typically large and customer requests are generated independently. We thus assume Poisson request arrivals. For simplicity, in most experiments we set the arrival rate equal to 40 requests/minute and the object playback duration to two hours for each object, and we vary the number of channels to examine relative policy performance as a function of offered load. We could equivalently have fixed the number of channels and varied the request arrival rate. Variable object playback durations are also considered in several experiments; in those cases we assume that objects have either short (fifteen minute) or long (two hour) playback durations, and we vary the fraction of requests to short and long objects as well as the number of channels.

The remaining assumption to be specified relates to customer reneging behavior in a competitive environment. We define the customer's tolerance to be the time that the customer is willing to wait for playback to begin. In this paper, we consider three different distributions of the customer tolerance, labeled A, B, and C, respectively, in figure 4. Note that the CDF of the tolerance is the probability that the customer tolerance is less than or equal to the given x value. For each distribution, the average customer tolerance (5 minutes) is given as a parameter, and for the normal distributions, the standard deviation is given as a second parameter. These tolerance models will be defined in greater detail when the results are presented in the next section.

5 Results

In this section, we compare the performance of the request scheduling policies, FCFS, FCFS-n, MFQ, and $GGSC_W$-FCFS, under the workloads described above. In the case of the FCFS-n policies, the number of channels reserved for each hot object allows a new playback of the object every 5 minutes; that is, $T = 5$ minutes. Unless otherwise specified, the experiments use the Zipf(0) distribution of object selection frequencies.

The simulators for FCFS and FCFS-n were validated by generating the curves in [11] and the MFQ simulator was validated by generating some of the results in [1]. Confidence intervals were generated by the method of batch means after discarding the initial transient. For the results given below, 95% confidence intervals were within 5% of the reported values.

We first consider competitive market environments and then we consider captive audience environments. Note that of all the policies we examine, only MFQ cannot provide the time of service to an arriving customer. This is a desirable system attribute because it makes customer waiting easier. Thus, if other performance metrics are approximately equal, FCFS or $GGSC_W$-FCFS would be preferred to MFQ.

5.1 Competitive Market Environment

5.1.1 FCFS and FCFS-n

All experiments in the section are based on the Zipf(0.271) distribution of object selection frequencies.

Figure 5 presents a performance comparison of the FCFS and FCFS-n policies. The figure shows, for each policy, the system cost as measured by the number of channels required, as a function of target customer loss probability.

In this experiment, the tolerance model is "5 or model B", meaning that customers wait if their time of service guarantee is less than or equal to 5 minutes; otherwise their tolerance is defined by distribution B in figure 4 (i.e., exponentially distributed with mean equal to 5 minutes). Note that in this tolerance model, which was used in [11], time of service guarantees that are under 5 minutes have a great impact on a customer's willingness to wait.

The most significant conclusion from figure 5 is that, particularly for system configurations with customer loss under 10%, the simple FCFS algorithm performs as well or better than the FCFS-10 algorithm. This is also true for FCFS-5, FCFS-20, or FCFS-40 (not shown), and for the other tolerance distributions considered in this paper (also not shown). The results thus argue that reserving capacity for the hot objects is not beneficial and may even be slightly detrimental to FCFS performance in high-end servers. Note that customer loss under 10% is highly desirable in a competitive server designed to capture a reasonable fraction of a given market.

These results and the conclusions regarding the viability of FCFS-n differ from previous work [11] primarily because we have simulated a FCFS policy that provides time of service guarantees to all customers, rather than only providing guarantees to customers that request objects with preassigned server capacity.

Another interesting observation from figure 5 is that, for the tolerance model that was simulated, only a small percent increase in

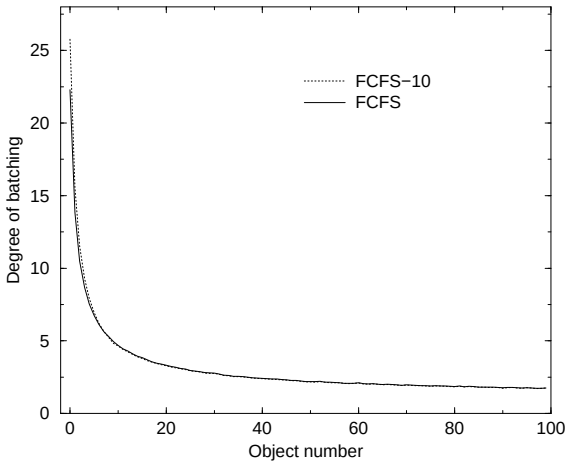
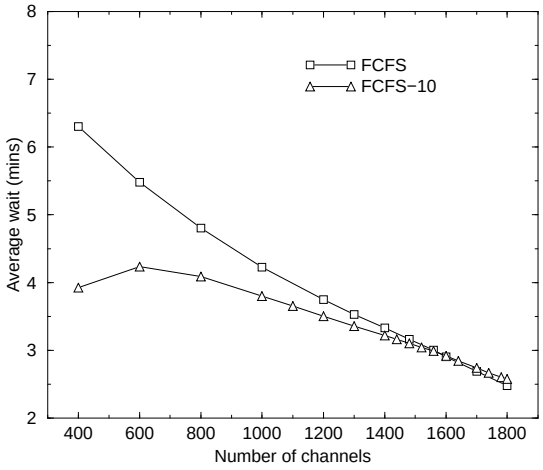
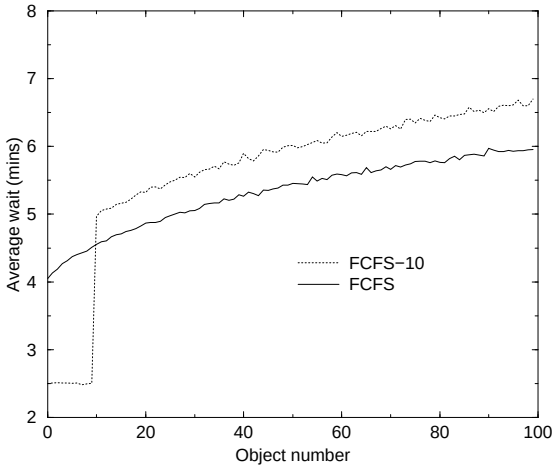


Figure 6: Degree of batching in FCFS and FCFS-n.
(tolerance model C; 1700 channels)



(a) Average wait v. # of channels.



(b) Average wait v. object popularity (800 channels)

Figure 7: Average wait for FCFS and FCFS-n.
(tolerance model C)

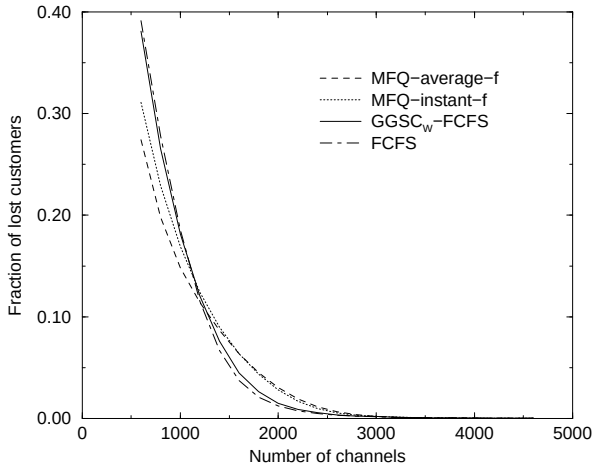


Figure 8: Customer loss v. number of channels.
(tolerance model A)

channels is required to decrease customer loss from 1% to 0.001%.

Figure 6 shows the average number of customers served in each playback versus object number in the FCFS and FCFS-n systems. In this experiment, tolerance model C is assumed; customers wait 5 minutes plus an additional time that is exponentially distributed with mean of 5 minutes. This distribution is probably more realistic than the tolerance model in figure 5. Note that significant batching occurs for popular objects in FCFS systems even though no objects have preassigned channels, and even under high-capacity system configurations that have extremely low customer loss.

Figure 7(a) shows that, at system configurations that have customer loss under 10%, the average wait for customers who don't renege is approximately the same for FCFS as for FCFS-n. As the number of channels decreases, or offered load per channel increases, the overall mean wait in FCFS is poorer than in FCFS-n; however, the undesirable bias of FCFS-n against the non-hot objects, shown in figure 7(b), also increases.

Given this set of results, we conclude that FCFS-n does not offer any significant advantages over FCFS, and thus we do not consider FCFS-n any further.

5.1.2 Comparison of FCFS, MFQ, and GGSC$_W$-FCFS

The FCFS, MFQ, and the newly proposed GGSC$_W$-FCFS algorithms are compared in this section for competitive server environments in which all objects have equal playback length. The results for environments with unequal object lengths are very similar and lead to the same conclusions about relative policy performance. For the sake of brevity and simplicity we give only the results for systems with objects of equal length.

Below we compare the policies for each of four different models of customer reneging behavior. In models A through C, customer tolerance is not affected by the time of service guarantees, whereas in model D tolerance is affected by the time of service guarantees.

Tolerance model A: In model A, customer tolerance follows the Normal(5, 1.67) distribution (see figure 4). This model was used in [1]; thus, in addition to its intuitive appeal, it enables comparison of our results with previous results.

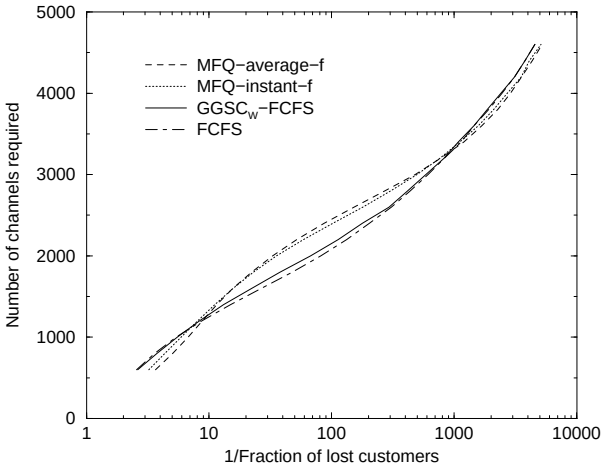


Figure 9: System cost v. target customer loss. (tolerance model A)

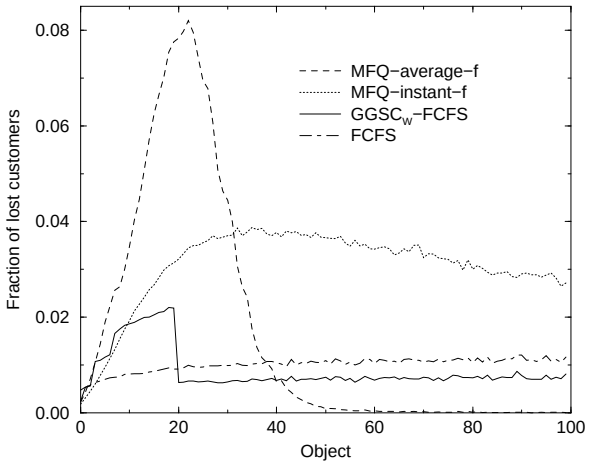
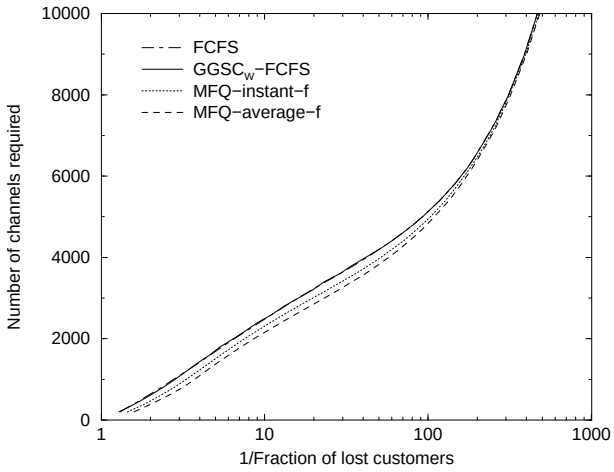


Figure 10: Customer loss v. object popularity. (tolerance model A; 2200 channels, FCFS loss = 1%)

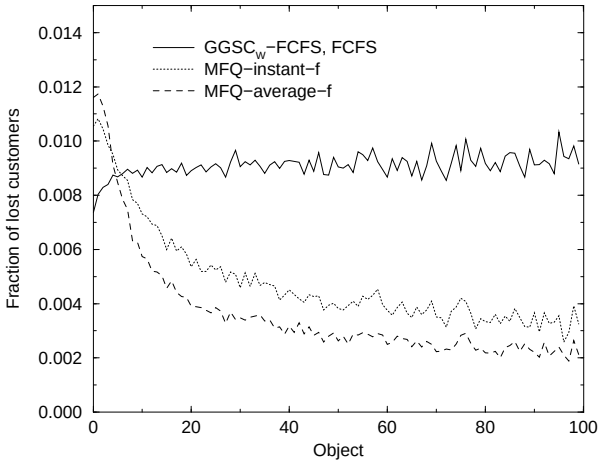
Figure 8 shows the fraction of customers lost as a function of the number of channels in the system. For configurations with customer loss above 10%, MFQ has lower loss than FCFS. This agrees with previous results in [1]. However, as discussed above, competitive systems will likely be configured for customer loss less than 10%. For such systems, FCFS has lower customer loss than MFQ, except for extremely low loss high-end configurations where the two algorithms have nearly equal cost. Note that GGSC$_W$-FCFS has performance very similar to FCFS.

A key question for a competitive system is: what is the required cost for a given (low) target customer loss? Figure 9 plots the cost (number of channels required) versus customers loss for the different request scheduling algorithms under tolerance model A. The x-axis is plotted on a log scale to focus attention on low loss configurations. This figure in fact contains exactly the same data as the previous figure 8, but in a form that better facilitates cost comparisons among the scheduling algorithms.

For target loss probabilities between 10% and 0.1%: (1) FCFS has lowest cost, (2) GGSC$_W$-FCFS has very similar cost (maximum difference in cost is less than 5%), but (3) MFQ has cost up to 20% greater than FCFS. For target loss less than 0.1%, figure 9 shows that all three systems have roughly equal cost.



(a) System cost v. target customer loss.



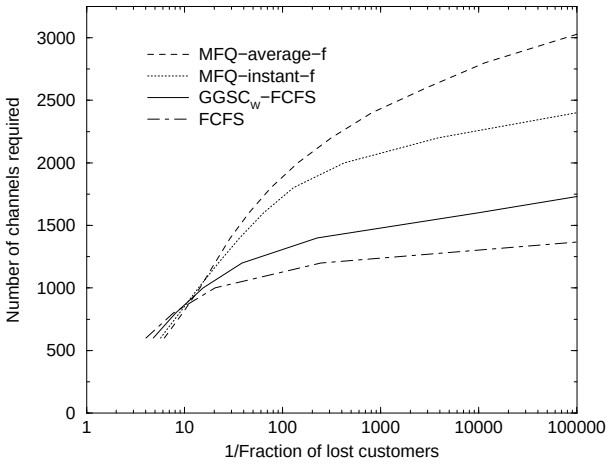
(b) Customer loss v. object popularity. (5400 channels)

Figure 11: Algorithm comparisons for tolerance model B

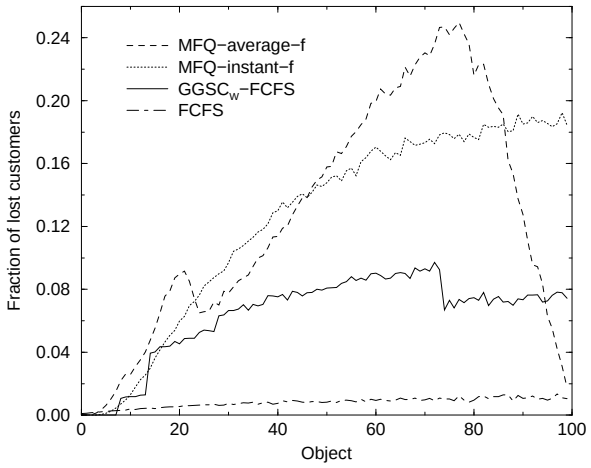
Figure 10 shows the fraction of customers lost for each object, for a configuration (2200 channels) where overall customer loss is about 1% in the FCFS system. Again, tolerance model A is assumed. As might be intuited, FCFS has nearly uniform loss across all objects; that is, nearly all objects have loss near the target (1%). The most popular objects have slightly lower loss since requests for those objects are somewhat more likely to be able to batch with a previous arrival. In MFQ, loss varies widely across objects (from 0.1% to 8%), with *many popular objects having loss that is two to eight times the target loss.* Finally, for GGSC$_W$-FCFS, loss varies from 0.5% to 2.8%; about 20% of the objects have loss that is two to three times the target loss.

Tolerance models B and C: In model B, customer tolerance has the exponential distribution with a mean of 5 minutes. In model C, all customers wait for at least 5 minutes; their tolerance above 5 minutes follows the exponential distribution with a mean of 5 additional minutes. These models were used in [11].

Figures 11(a) and 12(a) show the system cost versus customer loss for tolerance models B and C, respectively. For model B, FCFS and GGSC$_W$-FCFS have similar cost that can be up to 15% higher than MFQ, for target loss between 10% and 0.5%. However, for

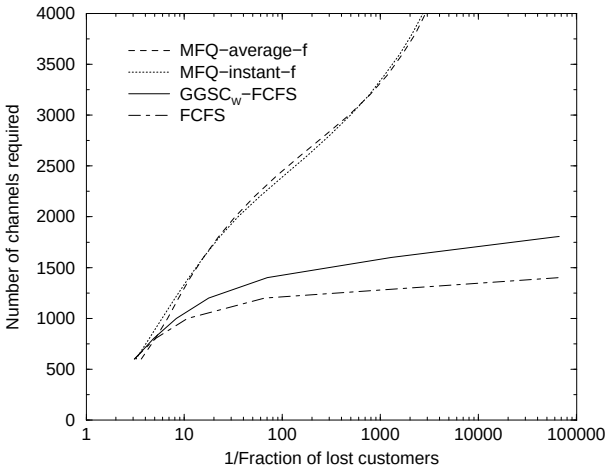


(a) System cost v. customer loss.

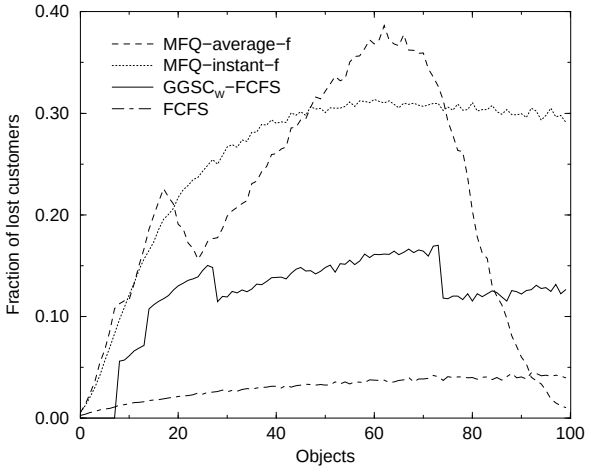


(b) Customer loss v. object popularity. (1200 channels)

Figure 12: Algorithm comparisons for tolerance model C.



(a) System cost v. customer loss.



(b) Customer loss v. object popularity. (1200 channels)

Figure 13: Algorithm comparisons for tolerance model D.

model C, FCFS has significantly lower cost than GGSC$_W$-FCFS, which is in turn far superior to MFQ for target loss lower than 10%. Note the difference between the results for tolerance models B and C; in model C all customers are willing to wait for a small threshold of time (5 minutes in this experiment), which results in FCFS, rather than MFQ, being the algorithm of choice. We hypothesize that tolerance model C, and perhaps also A, is closer to behavior that would be observed in practice than model B.

As observed for the two distributions of tolerance previously considered, FCFS has relatively uniform loss probability across the objects for model C, whereas GGSC$_W$-FCFS is less uniform and MFQ is highly variable (see figures 11(b) and 12(b)).

Note that tolerance models A, B, and C do not take into account that customers might be willing to wait longer if they are given a time at which playback will begin when they arrive. FCFS and GGSC$_W$-FCFS can give such time of service guarantees, whereas MFQ cannot. This is addressed in tolerance model D below.

Tolerance model D: In model D a customer who receives a time of service guarantee that is less than or equal to 5 minutes will wait to be served; otherwise the customer tolerance has the Normal(5, 1.67) distribution.

The results, shown in figure 13, are similar to the results for model C, but with a larger differential between MFQ and GGSC$_W$-FCFS and a smaller differential between GGSC$_W$-FCFS and FCFS. That is, if time of service guarantees influence some customers to wait a little longer, FCFS has the lowest cost for target loss under 10%, whereas MFQ has very high relative cost and the most highly variable loss versus object popularity.

5.1.3 Summary: Competitive Environments

Overall, the above results for competitive environments under a variety of tolerance models suggest that FCFS has the lowest cost and most uniform customer loss across objects of the algorithms considered in this paper. Note that this conclusion differs from prior work [1] primarily because we focus on systems with target loss under 10%. The results for systems that have unequal object playback times, not shown, are qualitatively the same with respect to relative performance of the scheduling policies.

5.2 Captive Audience Environments

We next compare the performance of FCFS, MFQ, and GGSC$_W$-FCFS for multimedia servers that have a captive audience, such as

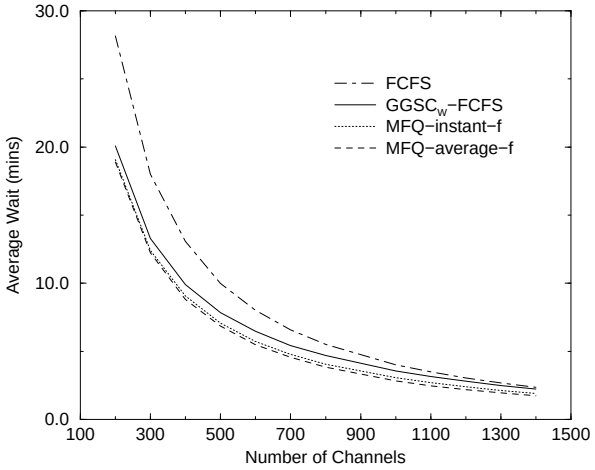


Figure 14: Average wait v. system cost.
Captive audience, uniform object playback lengths.

a campus or corporate server. In this case, the key performance metrics pertain to customer waiting time. We consider the average and *distribution* of customer waiting time as a function of system cost (number of channels) and as a function of object popularity. This range of metrics provides a more complete comparison of MFQ and FCFS than in prior work [1].

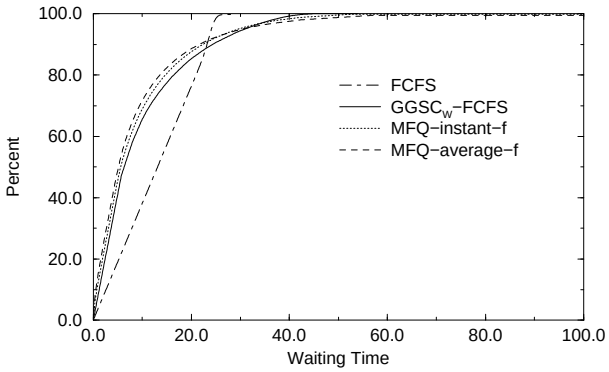
Section 5.2.1 considers the simple case where all objects have equal playback length; section 5.2.2 considers the perhaps more realistic case where objects have different playback lengths. The captive environment with unequal object lengths has not, to our knowledge, been investigated in earlier work.

5.2.1 Uniform Object Playback Lengths

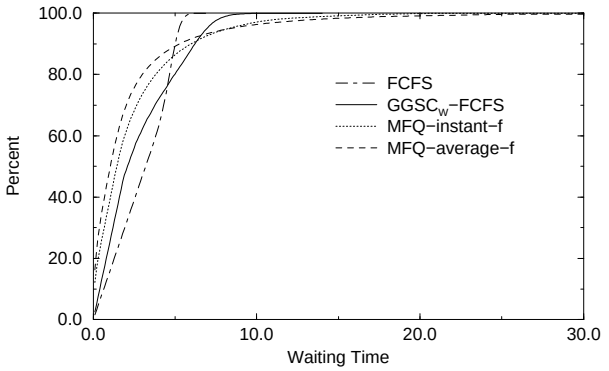
For the experiments in this section, all objects are assumed to have playback duration of 2 hours, as in the competitive environments studied above. The results should also hold for other uniform playback durations; performance is affected only by the relative magnitudes of the customer arrival rate, object playback durations, and system cost (number of channels).

Figure 14 shows the mean customer wait as a function of system cost. As in previous work, the MFQ policy has lower mean wait than FCFS throughout the range of system configurations. The difference is most significant for medium to low cost systems that have moderate to high load. On the other hand, the GGSC$_W$-FCFS policy has only slightly higher average wait than MFQ; at high load the mean wait is 6% higher for GGSC$_W$-FCFS while at low load the mean wait is about 30 seconds longer for GGSC$_W$-FCFS (e.g., about 135 seconds versus 105 seconds at 1400 channels).

To gain further insight into relative policy performance, figure 15 shows the distribution of the customer waiting times for two configurations, namely 400 channels (low cost, high load) and 1200 channels (high-end, light load). These curves show that (1) MFQ has the distribution with the heaviest tail, (2) FCFS has the lowest variability in wait time, and (3) GGSC$_W$-FCFS is in-between, although closer to MFQ for the low-cost configuration. For the low-cost configuration, average wait in MFQ is about 10 minutes but nearly one in twenty customers will wait longer than 40 minutes. FCFS has average wait closer to 13 minutes, but fewer than 7 in 10,000 customers wait more than 28 minutes. For the high-end configuration, the differences in average waiting time among the policies are small, and the lower variability in waiting time for FCFS or GGSC$_W$-FCFS is an even clearer advantage.

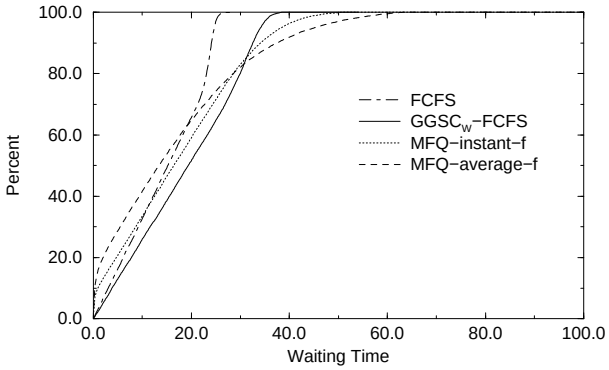


(a) Low cost, high load (400 channels).

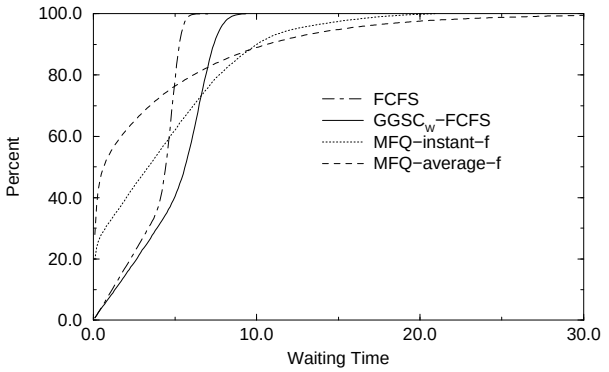


(b) High-end, low load (1200 channels).

Figure 15: Overall distribution of wait time.
(captive audience, uniform object playback lengths)



(a) Low cost, high load (400 channels).



(b) High-end, low load (1200 channels).

Figure 16: Median popularity object wait time distributions.
(captive audience, uniform object playback lengths)

	Light Load		Heavy Load	
	AVG Wait	MAX Wait	AVG Wait	MAX Wait
MFQ-average-f	2.2	144.5	8.8	193.9
MFQ-instant-f	2.4	33.1	9.1	136.7
$GGSC_W$-FCFS	2.8	14.0	9.8	53.7
FCFS	3.0	6.8	13.0	27.9

Table 1: Mean and maximum observed wait.
(captive audience, same length objects)

	All Objects		15-min Objects		2-hour Objects	
	AVG	MAX	AVG	MAX	AVG	MAX
MFQ-average-f	2.4	141.5	0.8	49.0	3.9	141.5
MFQ-instant-f	2.6	58.7	1.0	9.3	4.1	58.7
$GGSC_W$-FCFS	2.9	24.4	0.9	3.8	5.0	24.4
FCFS	4.6	10.0	4.6	10.0	4.6	10.0

Table 2: Mean and maximum observed wait for light load.
(captive audience; 100 (2-hour) and 100 (15-min) objects; 50% short object selection)

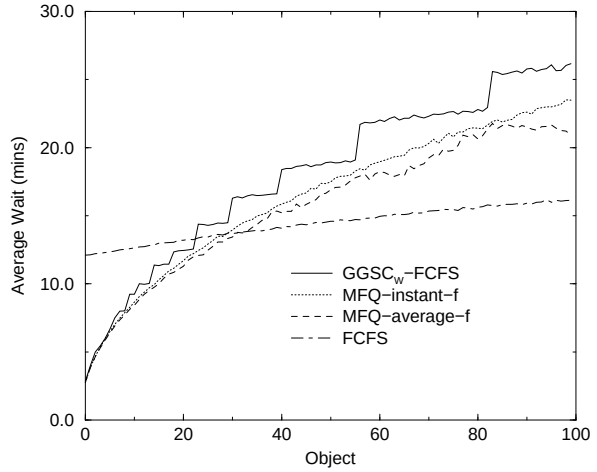


Figure 17: Average wait time versus object popularity.
(captive audience; uniform object playback lengths; low cost, high load (400 channels))

The waiting time distributions for the median popularity object in the low cost and high end systems, shown in figure 16, further illustrate the advantages of FCFS and $GGSC_W$-FCFS. All of the algorithms studied have low variability in wait time for the most popular objects; but variability in waiting time increases as object popularity decreases in MFQ.

Table 1 gives the average wait and the maximum observed wait for each algorithm for light and heavy load (1200 and 400 channels, respectively). From this table and the previous figures we observe that:

- at light load where the average wait is 2-3 minutes, the observed waiting times in MFQ-average-f were as large as 145 minutes (nearly two and a half hours!),
- MFQ-instant-f has significantly improved variability in waiting time compared with MFQ-average-f, yet still has waiting time up to 30 minutes for the system configured for average wait of 2.4 minutes, and
- FCFS appears to have the best performance characteristics when both average and distribution of waiting time are considered, both for low cost and high end systems that have uniform object playback durations.

Finally, the average wait as a function of object popularity is plotted for 400 channels (high load) in figure 17. For this low cost system configuration, FCFS offers significantly greater fairness in average waiting time, whereas $GGSC_W$-FCFS and MFQ give lower average wait for the most popular objects (i.e., better than 70% of customers) at the expense of greater average wait for the lower popularity objects. The differential in average wait across the objects for $GGSC_W$-FCFS and MFQ decreases as system cost increases; for example, at 1200 channels, the mean wait for the least popular object is only 25% higher under $GGSC_W$-FCFS than under FCFS.

We next consider the relative algorithm performance for captive audience environments where objects have unequal playback lengths.

5.2.2 Captive Audience, Nonuniform Object Playback Lengths

The system considered in this section has 200 objects, 100 of which are short (15 minutes long) and 100 of which are long (2-hours long). The selection frequency within either class, short or long, follows the same Zipf(0) distribution used in previous experiments. An incoming customer selects a long object with probability 50%; otherwise a short object is selected. Results for systems with (25%, 75%) and (75%, 25%) distribution of selection of long and short objects are omitted for the sake of space but show very similar relative performance of the algorithms.

The customer arrival rate is set such that offered load is equal to the offered load in previous experiments where all objects have 2 hour length and arrival rate equals 40 customers/minute (i.e., 40×120).

The graph of average wait versus number of channels, not shown, shows very similar relative policy performance as when all of the objects have equal length (see figure 14), but the gap between FCFS and the other algorithms is somewhat larger. We will see that this is due to the fact that requests for short objects have the same average wait as requests for long objects in FCFS.

Table 2 gives the mean and maximum observed wait for MFQ, $GGSC_W$-FCFS, and FCFS for a system with 1000 channels (high cost, short average wait).

From this table we observe the following:

- FCFS treats short and long objects identically, while both $GGSC_W$-FCFS and MFQ achieve significantly lower average waiting times for the short objects,
- overall, and for the objects with long playback time, FCFS has the lowest variability in waiting time; in fact, the distributions of the overall waiting time and the waiting time distributions for long objects, omitted for the sake of space, are similar to the waiting time distributions for equal length objects shown in figures 15 and 16,
- for the objects with short playback durations, the $GGSC_W$-FCFS and MFQ algorithms have the smallest average wait, and $GGSC_W$-FCFS has significantly better maximum wait; in contrast, FCFS has both high average wait and relatively high observed maximum wait; figure 18 further illustrates the superior performance of $GGSC_W$-FCFS and poor performance of FCFS for short objects.

In systems with unequal object playback lengths, it appears that $GGSC_W$-FCFS has some clear performance advantages over FCFS and MFQ.

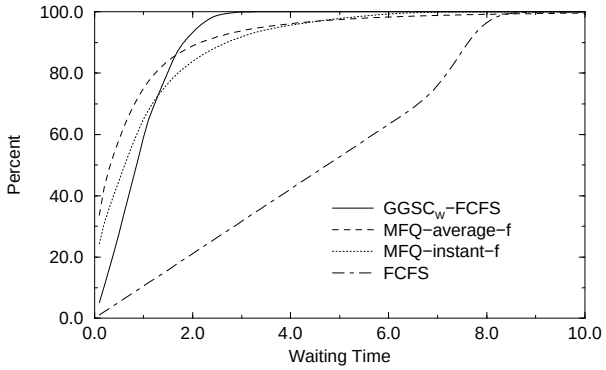


Figure 18: Distribution of waiting time for short objects. (captive audience; 100 (2-hour) and 100 (15-min) objects; 50% short object selection; light load (1000 channels))

Figure 19 shows the average wait as a function of object popularity for both long and short objects, under light load (1000 channels). The three top curves are for long objects under GGSC$_W$-FCFS, MFQ-instant-f, and MFQ-average-f, respectively; the middle curve is for short and long objects under FCFS; the bottom three curves are for MFQ-instant-f, GGSC$_W$-FCFS, and MFQ-average-f. Note that for this high-end system, customers of the least popular 2-hour object wait, on average, for about twelve minutes in GGSC$_W$-FCFS versus six minutes in FCFS, but customers requesting any of the 15-minute objects have average wait of one to two minutes in GGSC$_W$-FCFS versus four to six minute in FCFS. MFQ has perhaps even better mean wait as a function of object popularity than GGSC$_W$-FCFS, but MFQ is not viable due to the tail of its waiting time distribution for long objects.

For the captive audience environment with unequal object lengths we can conclude the following:

- FCFS has the lowest variability of wait times but the average wait is disproportionally high for customers of short objects, and
- GGSC$_W$-FCFS provides waiting times that have small mean and low variance relative to the playback length of the requested object, but the average wait is relatively nonuniform across objects of similar length but differing popularity.

6 Conclusions

In this paper, we have introduced the Group-Guaranteed Server Capacity (GGSC) family of request scheduling algorithms for multimedia storage servers. These policies preassign server capacity to groups of objects and schedule requests within a group in FCFS order. The capacity assignments can be optimized for a specific performance objective; yet arriving requests can be given time of service guarantees. As a special case we defined the GGSC$_W$ algorithm that minimizes the average waiting time when the system operates at full capacity.

We compared the performance of FCFS, the previously proposed FCFS-n and MFQ algorithms and the new GGSC$_W$-FCFS algorithm under two particular environments: competitive market and captive audience. In the competitive market environment, the principal metric used to compare the policies was system cost required to achieve a given target customer loss probability. We also considered customer loss as a function of object popularity. For the customer tolerance models we considered, the FCFS policy has both the best cost performance and the best uniformity in customer loss per object.

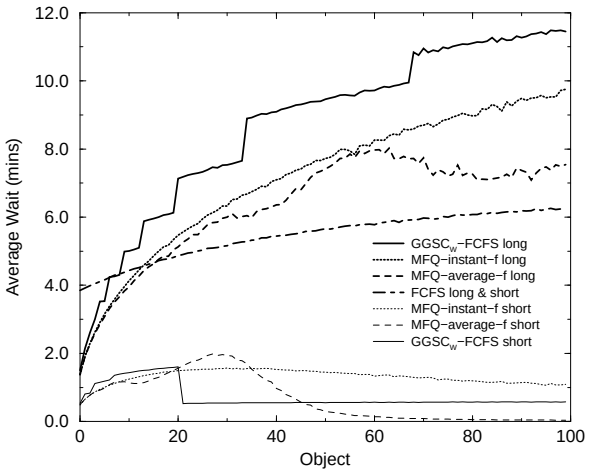


Figure 19: Average wait versus object number. (captive audience; 100 (2-hour) and 100 (15-min) objects; 50% short object selection; light load (1000 channels))

In the captive audience environment, we looked at mean and distribution of customer waiting time as a function of system cost and of object popularity. In systems with unequal object playback lengths, the GGSC$_W$-FCFS policy provides waiting times that have small mean and low variance relative to the playback length of the requested object and appears to be the most promising policy.

The principal reasons that our conclusions regarding FCFS-n and MFQ differ from prior work are: (1) we assume that FCFS offers time of service guarantees to incoming customers, (2) we focus on competitive environments with customer loss probability less than 10%, and (3) we considered distribution of waiting time as well as the mean waiting time in assessing the relative merit of the algorithms.

Our current research includes investigating (1) GGSC algorithms that use minimizing customer loss as the objective function for channel assignments, (2) alternate algorithms for scheduling requests when a channel becomes free and no requests are waiting for objects in its group, and (3) the interactions between GGSC algorithms and disk scheduling and buffering for multimedia servers.

References

[1] AGGARWAL, C. C., WOLF, J. L., AND YU, P. S. On optimal batching policies for video-on-demand storage servers. In *Proceeding of the IEEE International Conference on Multimedia Computing and Systems* (Hiroshima, Japan, June 1996).

[2] ALMEROTH, K., DAN, A., SITARAM, D., AND TETZLAFF, W. Long-term channel allocation strategies for video applications. Tech. Rep. GIT-CC-95-45, RC 20249, Georgia Tech, IBM Research Report, Yorktown Heights, NY, 1995.

[3] BERSON, S., GOLUBCHIK, L., AND MUNTZ, R. R. Fault tolerant design of multimedia servers. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (San Jose, CA, USA, May 1995), pp. 364–375.

[4] BERSON, S., MUNTZ, R., GHANDEHARIZADEH, S., AND JU, X. Staggered striping in multimedia information systems. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data* (Minneapolis, Minnesota, USA, May 1994), pp. 79–90.

[5] DAN, A., DIAS, D. M., MUKHERJEE, R., SITARAM, D., AND TEWARI, R. Buffering and caching in large-scale video servers. In *Proceedings of IEEE CompCon* (San Francisco, March 1995), pp. 217–224.

[6] DAN, A., KIENZLE, M., AND SITARAM, D. A dynamic policy of segment replication for load-balancing in video-on-demand servers. *ACM/Springer-Verlang Multimedia Systems Journal 3*, 3 (1995).

[7] DAN, A., SHAHABUDDIN, P., SITARAM, D., AND TOWSLEY, D. Channel allocation under batching and VCR control in movie-on-demand servers. *Journal of Parallel and Distributed Computing 30*, 2 (November 1995), 168–179.

[8] DAN, A., AND SITARAM, D. Buffer management policy for an on-demand video server. Tech. Rep. RC 19347, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 1993.

[9] DAN, A., AND SITARAM, D. An online video placement policy based on bandwidth to space ratio (BSR). In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (San Jose, CA, USA, May 1995), pp. 376–385.

[10] DAN, A., AND SITARAM, D. A generalized interval caching policy for mixed interactive and long video workloads. *Multimedia Computing and Networking* (January 1996). IBM Research Report RC 20206.

[11] DAN, A., SITARAM, D., AND SHAHABUDDIN, P. Scheduling policies for an on-demand video server with batching. In *ACM Multimedia '94* (San Francisco, CA, USA, October 1994).

[12] *Electronic Engineering Times* (March 15 1993), 72.

[13] FREEDMAN, C. S., AND DEWITT, D. J. The SPIFFI scalable video-on-demand system. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (San Jose, CA USA, May 1995), pp. 352–363.

[14] GHANDEHARIZADEH, S., KIM, S. H., AND SHAHABI, C. On configuring a single disk continuous media server. In *Proceedings of the 1995 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (Ottawa, Ontario, Canada, May 1995), pp. 37–46.

[15] GOLUBCHIK, L., LUI, J. C., AND MUNTZ, R. Reducing I/O demand in video-on-demand storage servers. In *Proceedings of the 1995 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (Ottawa, Ontario, Canada, May 1995), pp. 25–36.

[16] KAMATH, M., RAMAMRITHAM, K., AND TOWSLEY, D. Management for continuous media sharing in multimedia database systems. Tech. Rep. UM-CS-1993-081, Department of Computer Science, University of Massachusetts at Amherst, Amherst, MA 01003, March 1994.

[17] NG, R., AND YANG, J. Maximizing buffer and disk utilization for news on-demand. In *Proceedings of VLDB* (1994).

[18] PERRY, T. S. The trials and travails of interactive TV. *IEEE Spectrum* (April 1996), 22–28.

[19] TRISTRAM, C. Stream on: Video servers in the real world. *Newmedia* (April 1995), 46–52.