Scheduling Policies for an On-Demand Video Server with Batching

Asit Dan
Dinkar Sitaram
Perwez Shahabuddin

IBM Research Division, T. J. Watson Research Center
Yorktown Heights, NY 10598

ABSTRACT

In an on-demand video server environment, clients make requests for movies to a centralized video server. Due to the stringent response time requirements, continuous delivery of a video stream to the client has to be guaranteed by reserving sufficient resources required to deliver a stream. Hence there is a hard limit on the number of streams that can be simultaneously delivered by a server. The server can satisfy multiple requests for the same movie using a single disk I/O stream by sending the same data pages to multiple clients (using the multicast facility if present in the system). This can be achieved by batching requests for the same movie that arrive within a short duration of time. In this paper, we consider various policies for selecting the movie to be multicast. The choice of a policy depends very much on the customer waiting time tolerance before reneging. We show that an FCFS policy that schedules the movie with the longest outstanding request can perform better than the MQL policy that chooses the movie with the maximum number of outstanding requests. Additionally, if the user behaviour can be influenced by guaranteeing maximum waiting time then it may be beneficial to pre-allocate a fixed number of streams for popular movies. Finally, we demonstrate using empirical distribution for movie requests, that a substantial reduction (more than 60%) in required server capacity can be achieved by batching.

IBM Research Report, RC 19381

1 Introduction

Recent advances in communication and computer technology have made feasible video-on-demand applications, where a large database of movies are stored in a set of centralized servers and played through high-speed communication networks by geographically distributed clients (see Figure 1) [6, 15, 14]. Due to the stringent response time requirements, continuous delivery of a stream to the client has to be guaranteed by reserving sufficient resources (both at the network as well as at the server) required to deliver a stream [1, 14]. The bottleneck resource at the server could be either disk bandwidth or CPU capacity. Hence there is a hard limit on the number of streams that can be simultaneously delivered by a server.

Each client requests playback of a movie of his/her choice independently of others. Such a request may arrive at random time intervals. Hence a new movie stream could be started to satisfy the request. In a commercial environment, there may be a large number of clients simultaneously watching movies over the network, potentially in the thousands [5]. This would require a very large server capacity. We would like to note that modern communication networks such as ATM are equipped with a multicast facility [12, 9, 22] i.e. the same message can be sent to multiple clients without causing any extra overhead to the server. This feature can be exploited to reduce the number of streams required by a server to support a given number of clients. For example, if two clients make a request for the same movie separated by a small time interval, then by delaying the playback for the first client, the same server stream can be used to satisfy both requests [1]. The multicast facility need not be present in the distribution system. A video server that reads once from disk and then sends separately the data to multiple clients accrues the benefit of reading once for multiple clients. In general, multiple clients request for the same movie within a short time duration can be batched together and serviced using a single stream.[1] This is referred to as *merging* in this paper. In the following, we explore

[1]In this paper, we do not consider the problem of providing VCR-like controls such as pause and

various scheduling policies that arise as a result of this property.

In general, increasing the batching window reduces the required server capacity while also increasing the average client waiting time. Clients may not always be willing to wait a longer time before being served and may cancel (*renege* in queueing theory terminology) their requests. Thus there is a tradeoff between decreasing server capacity and increasing both waiting time and reneging probability. An additional factor that determines the average client waiting time is the choice of a movie to be played at any given time. The policy that chooses a movie that has the largest number of waiting clients attempts to minimize the number of lost customers. However, this policy may not be fair since it does not pick the client who has been waiting the longest. Hence various policies arise out of the tradeoff amongst these factors. The maximum waiting time of a client before it cancels its request (referred to as *reneging time*) itself can be influenced by the scheduling policy. For example, a client may not want to wait longer under uncertainity. However, guaranteeing a maximum waiting time by starting movies at predetermined intervals even for only one viewer, can change client behavior so as to increase its reneging time. Since not all movies are requested with the same frequency, such a guarantee may be appropriate only for certain movies. These interactions between client behaviour and scheduling policies are also studied in this paper.

The queueing systems studied in this paper are distinguished from those in earlier works primarily by the combination reneging and merging property. The non-work conserving merging property appears as a result of the multicast feature in multimedia systems. This has also been observed in the context of broadcast delivery in videotex systems [20, 4]. In such a system, users submit independent requests to retrieve a specific page

restart for individual clients. There are various ways to provide this facility. For example, if the client was watching a multicast movie stream, it is likely that there will be another multicast stream for such a popular movie following shortly that the client can join. An alternative approach is to use a small amount of buffer for caching the blocks of a multicast stream for the paused clients similar to [3].

of information to a service computer. When the service computer processes a request the desired page is broadcast to all users regardless of who requested it. Our system differs from the videotex system in several ways. First, the service time in the videotex system consists only of retrieving a page. (Broadcasting of a page is istantaneous.) Hence, all requests for that page that arrive until the instant of broadcast are served. In contrast, the service (broadcast) time for a movie is very long. Additionally, there are multiple channels for serving movie requests in our system. Finally, the scheduling policy in our system has to take into account the customer reneging behavior.

Loss systems have been studied both in the classical literature [8] and also in various communication systems [13, 21]. The loss may occur due to two reasons. In communication systems, the loss may occur due to finite resources (e.g., buffer) and hence, the constraint on the queue length. In some other queueing systems the loss may be due to reneging of impatient customers [7, 21] or due to missing deadline in hard real-time environment [11, 16, 17, 2]. In our system, the individual client deadlines are not known, and the loss is primarily due to reneging of impatient clients. The work closest to ours in this respect is the paper by Zhao, Panwar and Towsley [21] where the behavior of individual impatient customers is not known. Apart from the merging property, the current work differs from the above paper in many respects. In addition to minimizing reneging probability, we also explore the fairness issue for all request types and interaction between the scheduling policy and customer behavior.

The remainder of the paper is as follows. In section 2, we propose several scheduling policies and discuss various performance tradeoffs associated with these policies. Section 3 attempts to model these policies analytically and demonstrates the inherent complexity that arises from the merging property. Hence the remainder of the paper concentrates on studying these policies through simulation. The tradeoffs amongst various policies are studied in section 4. Section 5 contains our concluding observations.

2 Assumptions and Proposed Policies

Access to the movies are non-uniform, i.e., some movies are more popular than others. Figure 2 shows the frequency of rentals for various movies in a particular week in video stores [5]. It is interesting to note that the access frequencies to various movies can be characterized by a Zipf distribution, i.e., if the movies are sorted according to the access frequency then the access frequency for the i^{th} movie is given by, $f_i = c/i^{(1-\theta)}$, where θ is the parameter for the distribution and c is the normalization constant. The access skew changes with the parameter θ. A uniform distribution corresponds to the value of 1, and a value of 0 represents a highly skewed distribution. The value of θ for this particular example is 0.271.

Popular (hot) movies can benefit substantially by batching multiple requests since a large number of requests may arrive within a short time. The unpopular (cold) movies on the other hand do not benefit from making a customer wait since it increases the waiting time of the already arrived customers and there is a very little chance of new requests arriving for the same movies. Therefore, fixed batching intervals for all movies is not appropriate. A longer waiting time may also cause a customer to leave. Satisfying all requests may require a large server capacity. Hence, a system may be configured to satisfy only a certain fraction (say 95%) of requests. The various policies therefore differ in the choice of a movie to be multicast at any given moment. A policy may choose to multicast only those movies that satisfy the maximum number of requests. Such a policy may not be fair since it may never serve requests for very cold movies. Therefore, all policies are guided by three primary objectives:

- **Minimize Reneging Probability**[2]: This is a very important objective since a policy that minimizes reneging probability for a given server capacity would require

[2] In queueing theory, a customer is said to renege if it leaves the queue before being served. In the current context, this refers to a client that cancels its request due to excessive waiting time.

a lower server capacity to satisfy a given acceptance rate.

- **Minimize Average Waiting Time:** A secondary objective of a policy is to minimize average waiting time and/or variance in the waiting time of the clients that are served.

- **Fairness:** An orthogonal objective of all policies is to maximize fairness over requests for all movie types. Intuitively, fairness is defined as equal reneging probability for all movie requests.

Note that the number of requests for a particular cold movie may be very small during any time period. Therefore, the reneging probability for a particular cold movie and hence, unfairness may be difficult to estimate. The movies can be classified into bins, such that movies in each bin receive roughly the same number of requests. We now define a measure of unfairness, as

$$U = \frac{\sum_{i=1}^{M}(r_i - r_{av})^2}{M}, \tag{1}$$

where M is the number of bins, r_i is the reneging probability of the i^{th} bin movies and r_{av} is the average reneging probability over all bins. Note that unfairness is zero if movies in all bins have the same reneging probability, and unfairness increases as the values of r_i deviate from that of r_{av}. In the example in Figure 2, the movies are divided into 10 bins such that roughly 10% of the requests fall in each bin. We will use this workload for the simulation study of the scheduling policies in the next section.

2.1 Proposed policies

We propose two orthogonal classes of policies that select a movie for multicasting based on either the number of waiting customers or based on the customer that has been in the system for the longest time. Alternatively, policies can be devised that take into account both the waiting time as well as the number of customers (e.g., sum of waiting time for all customers for a movie as in [4]). Such policies are more complex and may not be preferred from the implementation point of view. We will not consider such policies in this paper.

- **FCFS Policy:** Under the *First Come First Served policy* the requests for all movies join a single queue which we call the requests queue. Each customer may leave the queue independently of others based on its reneging time. Once the server capacity for delivering a stream becomes available,[3] the client at the front of the requests queue is served. Note that because of the multicast facility, all customer requests for the same movie are also satisfied by the same stream. This non-work conserving property which we call *merging* makes it an interesting queueing theory problem. Intuitively, FCFS seems a fair policy since it selects a movie request independent of the identity of the movie.

- **MQL Policy:** Under the *Maximum Queue Length policy,* requests for each movie join a separate queue, and the movie with the maximum queue length is selected for multicast. This is similar to the MRF (Maximum Request First) policy in [4]. One drawback of this policy is that it may choose only the hot movies since there are very few requests for cold movies within a short time period (within the reneging time of a customer). Hence, this will considerably increase the reneging probability of the requests for the cold movies causing an increase in unfairness. However, the policy can better utilize a small server capacity to reduce the overall reneging probability.

- **FCFS-n Policy:** This is similar to the FCFS policy except that a fraction of the server capacity is reserved and pre-allocated for batching requests for the n hottest

[3] In this paper, we assume a steady-state in arrival rate. In a transient situation, clients may be delayed even if capacity is available so as not to exhaust the server capacity with the first few requests and to be allow batching.

movies. For all the movies with dedicated streams, a new stream is started every B minutes (referred to as batch interval). The policy may be somewhat unfair towards the cold movies. However, there are many advantages of pre-allocation of server capacity for the hot movies. First, a guarantee on the maximum waiting time can be provided for the request for the popular movies. Such a guarantee can be used to influence the reneging behavior of a customer. Second, the policy guarantees a high acceptance probability with a relatively small server capacity as the requests for the cold movies do not interfere with the services for the hot movies. Note however, that a large value of n may be inefficient since a lot of reserved capacity for the movies that are not so hot will be underutilized.

The remaining cold movies are served according to the FCFS policy, i.e., FCFS-0 is the same as the pure FCFS policy. If no movie requests arrive for a particular reserved stream for a hot movie, the stream can be used to multicast a cold movie. Note also that in a similar manner we can define MQL-n policy where a certain amount of server capacity is reserved for the n hottest movies, and the remaining movie requests are served according to the MQL policy. However, such a policy may not be very interesting since the MQL policy already favours the hottest movies.

2.2 Customer reneging behavior

The scheduling policy as well as the decision to pre-allocate server capacity for certain movies also depend on the user behavior. The amount of time an user will wait before deciding to leave may not be known in advance as is the case in deadline driven scheduling [11, 16, 17, 2]. In general, the reneging time of a client, which we denote by W, may be a random variable with a general distribution. If the probability that a customer leaves at any moment is independent of the amount of time the client has been in the system (this happens if W is exponentially distributed), and if the clients for all movies have the same mean reneging time, then all customers in the queue are equally likely to remain. Therefore, selecting a movie for multicasting to satisfy certain objectives becomes simpler. For example, the MQL policy always performs better than the FCFS policy in terms of minimizing the overall reneging probability. Similarly, simpler policies can be devised to satisfy alternative objectives like minimizing unfairness etc. However, despite the analytical simplicity of the exponential assumption, real people are unlikely to behave that way. People will more likely to renege as the delay in starting a movie increases. If the residual reneging time is dependent on the amount of time the client has been in the system, then policy selection becomes more difficult. The customer waiting time can also be influenced through prior negotiation, and this knowledge can be successfully exploited in the scheduling of movies. In this paper, we will assume two different models of customer reneging behavior.

- **Minimum Waiting Time:** Under this model, each customer is willing to wait at least W_{min} amount of time. The remaining waiting time is assumed to be exponentially distributed (i.e., $W = W_{min} + E$, where W_{min} is a constant and E is a exponentially distributed random variable. Hence, W_{min} could be used as the duration of a batching interval for the hot movies if the server capacity is preallocated. This will make the reneging probability of the client for the dedicated movies equal to zero.

- **Maximum Waiting Guarantee:** Under this model, a customer does not agree to a minimum waiting time. However, customers agree to wait if a maximum waiting time, W_{max}, is guaranteed. This is achieved by preallocating enough server capacity for those movies such that a new stream is started every W_{max} amount of time.

3 Markov Modelling of Video on Demand Systems

In this section we will outline some Markov modelling methods for some simple video on demand systems. We will see that even for these simple systems, these methods are prohibitive, due to the large state space.

First of all we assume that there are no dedicated service streams, and any movie may be served by any service stream. The two disciplines we will consider are MQL and FCFS, both of which have been described in the previous section. For each discipline, we will first consider the case with no reneging. Though the case with no reneging is not of much interest in our general setting, it gives us a starting ground to go on to the more difficult problems where we have reneging.

Let there be m movies with λ_1, λ_2, $\ldots$, λ_m being the rate of requests for the m movies. Assume that we have d service streams and all movies are of constant length L. We assume that the playing of movies in these service streams is evenly staggered, i.e. on the ith channel, movies (if any requests are waiting) start only at times $(iL/d, L+iL/d, 2L+iL/d, \ldots)$. This means that every $D = L/d$ time units we have the capability of starting exactly one movie.

3.1 FCFS with no reneging

In this case since there are no requests reneging, the quantity of interest is the expected waiting time for requests for each movie. Before the beginning of each window, there may be many requests for different movies waiting. In FCFS, which movie will be shown in the next window depends on the order of arrival of the waiting requests. If the first request in the requests queue is for movie i then all requests for movie i will be taken care of in the next window, and the system will be emptied of movie i requests. Hence the future evolution of the system depends only on the order of arrival of the first request for each movie in the requests queue. For example, if in a system with 5 movies, the requests queue is $(3,3,2,3,1,2)$, all we need to know is that the order of arrival of the first request for each movie is $(3,2,1)$. Hence define the state of the system to be $(j_1, j_2, \ldots, j_k)$ where k is the number of different movies for which requests are waiting (it excludes the movies which are being played) and $(j_1, j_2, \ldots, j_k)$ gives the order of arrival of the first requests for each of these k movies. We call $(j_1, j_2, \ldots, j_k)$ the *movie queue*, differentiating it from the requests queue. Then the random sequence of movie queues sampled at the start (or more precisely, immediately after the start) of successive windows, constitutes a Markov chain, i.e., the future evolution of this sequence depends only on the movie queue at the start of the current window.

In principle it is easy to determine the transition matrix for this Markov chain. For example the probability of going from $(3,2,1)$ to $(2,1,4)$ is the probability that at least one request for movie 4 arrives in the window interval of length D, and no request for movie 5 arrives in this time. Whether requests for movies 1, 2, or 3 arrive or not, is immaterial as they are already represented in $(3,2,1)$. Hence the probability of going from $(3,2,1)$ to $(2,1,4)$ is $(1-e^{-\lambda_4 D})e^{-\lambda_5 D}$. Once we compute the transition matrix, then we can solve for the $\pi(j_1, j_2, \ldots, j_k)$'s, which is the steady state probability of being in state $(j_1, j_2, \ldots, j_k)$.

Once we determine the steady state probabilities, then using the Poisson Arrivals See Time Averages (PASTA) property [19], we can determine the expected waiting time of requests for any given movie. This property basically states that, if the arrivals are Poisson, then the long run fraction of requests that arrive when the system in state $\mathbf{j} = (j_1, j_2, \ldots, j_k)$, is given by $\pi(\mathbf{j})$. The next step is to determine the expected waiting time of a request for movie i, arriving in state $(\mathbf{j})$ which we denote by $w_i(\mathbf{j})$. Then $E(W_i) = \sum_{\forall \mathbf{j}} w_i(\mathbf{j})\pi(\mathbf{j})$. Determining $w_i(\mathbf{j})$ in closed form is not difficult. For example, a request for movie 2 that arrives during a window that started in state (3,1,2), will have to wait till the end of the current window and two more windows, before it's service starts. (It is well known that the time till the end of the current window, will be uniformly distributed in $[0, D]$). A request for movie 4 that arrives in $(3, 1, 2)$ will have to wait till the end of the current window, three more windows in which movies 3,1,1 are played, and maybe a window where movie 5 is played (if movie 5, arrives before it in the current window; the probability of this event can easily be computed).

The approach outlined above may work in principle, but in practice it is infeasible, as the complexity of the problem is combinatorial in the number of movies. Note that the size of state space is approximately $\sum_{k=0}^{m-1} m!/(m-k)!$. This is because, the number of movies which can be in the movie queue, can vary from from 0 to $m-1$. Also, there are $m!/((m-k)!k!)$ ways of choosing k movies out of m movies, and there are $k!$ ways of ordering these k movies.

3.2 FCFS with reneging

Now consider the case where we have reneging. We will assume that each customer waits in the requests queue for a random amount of time W (that is not known in advance), that is exponentially distributed with rate λ_r. If the request is not served in this time, then it reneges. The exponentially distributed assumption implies that the time to renege has a memoryless property. Cases where W is other than exponentially distributed are considerably harder, and in most cases not amenable to Markov modelling.

We see now that just keeping the order of arrival of the first request for each waiting movie is not enough. Which movie will start service in the next window, depends on the entire request queue. For example, there is a much greater chance of movie 3 being served in the next window, if at the start of the current window the requests queue is $(3, 3, 2, 3, 1, 2)$ than $(3, 2, 3, 1, 2)$ (both of which have the same order of arrival of 1st request of each waiting movie), as the chance of the 1st two 3's reneging is less than the chance of one 3 reneging. However, since the wait is memoryless, defining the state to be the requests queue is enough to get a Markov chain. Note that now the state space goes from finite to infinite. If we use truncation on the arrival distribution (see Section 3.3).

then in principle it is possible to determine in closed form the transition probabilities of the Markov chain (and thus, solve for the steady state probabilities and consequently, using PASTA property, solve for the fraction of each class that reneges and the expected waiting times for those served).

3.3 MQL with no reneging

In this case we have to know how many requests for each of the movies is waiting, at the beginning of each window, i.e., the future evolution of system depends only on this information. Hence the state of the Markov chain is given by $(n_1, n_2, \ldots n_m)$, where atleast one of the n_is is zero (corresponding to the movie which started playing at this window). Note that this is an infinite state Markov chain, and unless a suitable structure is found, it may be impossible to solve. However one may be able to get good approximations by truncating the number of requests for each movie that arrive in a window. In the original system, since the arrivals are a Poisson process, the probability of k requests for movie i is given by $p_i(k) = e^{-\lambda_i D}(\lambda D)^k/k!$. In the truncated system we select a $\bar{k}_i$ such that $1 - \sum_{k=0}^{\bar{k}_i} e^{-\lambda_i D}(\lambda D)^k/k! \approx 0$. Then the new probability of k requests for movie i, is defined to be $\tilde{p}_i(k) = (e^{-\lambda_i D}(\lambda D)^k/k!)/(\sum_{j=0}^{\bar{k}_i} e^{-\lambda_i D}(\lambda D)^j/j!)$ for $0 \leq k \leq \bar{k}_i$, and zero otherwise.

Having truncated the state space it is easy to determine the transition matrix for the Markov chain. For example, in the five movie example, the probability of going from $(3, 0, 7, 5, 0)$ to $(0, 0, 9, 5, 3)$ is given by $(\sum_{i=6}^{\infty} \tilde{p}_1(i))\tilde{p}_2(0)\tilde{p}_3(2)\tilde{p}_4(0)\tilde{p}_5(3)$. Then one can solve for the steady state probabilities. Having determined the $\pi(n_1, n_2, \ldots, n_m)$'s, one can again use the PASTA property to determine the waiting time of the requests for the ith movie. However, in this case also, the growth of the state space size is exponential in the number of movies. For example in a system with m movies, the size of the state space is approximately $\prod_{i=1}^{m}(2\bar{k}_i)$.

3.4 MQL with reneging

Finally, let us consider the MQL case with reneging. As in the no reneging case, it is enough to define the state of the system to be the number of requests for each movie, waiting at the start of any window. If $(n_1, n_2, \ldots, n_m)$ were waiting at the start of the window, then by the end of that window, only $n'_1, n'_2, \ldots, n'_m$ will remain were n'_i is binomially distributed with parameters n_i and $e^{-\lambda_r D}$. In addition, requests will arrive in this window, and a fraction of them will leave before the window ends. Note that, given that k arrivals from a Poisson process have occurred in an interval, each of these arrivals times is uniformly distributed in the interval. Hence the probability of an arrival in a window, staying till the end of the window is given by

$$f(\lambda_r, D) = \int_{s=0}^{D} e^{-\lambda_r(D-s)} \frac{ds}{D}. \tag{2}$$

Then it is easy to see that the number of requests of movie i that arrive in the current window, and stay till the end of the window is a Poisson random variable with parameter $\lambda_i f(\lambda_r, D)D$; denote this random variable by N'_i. Which movie will be served in the next window will thus be determined by comparing the $(N'_i + n'_i)$'s for $1 \leq i \leq m$ and the successive state will be determined accordingly. Hence, using state space trucation we can again formulate a finite state space Markov chain, the transition probability of which can be (in principle) determined. After solving for the $\pi(n_1, n_2, \ldots, n_m)$'s we can find the long run fraction of requests for a given movie that renege. The size of the (truncated) state space will the same as in the no-reneging model.

3.5 MQL with general reneging times

Some Markov modelling is possible in the case of generally distributed reneging time random variable W (denote the general distribution by $G(\cdot)$), that is bounded above by a constant with probability 1. For example consider the case where the reneging time is bounded above by $3D$ w.p. 1. In this case, instead of only storing the total number of requests for each movie waiting, we have to differentiate between how many of these requests came in the previous window, and how many in the window before that (requests that came in three windows before may be ignored, as they would not wait till the end of the current window). Hence the state at the beginning of a window is given by $(n_{1,1}, n_{2,1}, \ldots n_{m,1}; n_{1,2}, n_{2,2}, \ldots n_{m,2})$, where $n_{i,j}$ is the number of movie i requests that came in j windows before (note that $n_{i,1} + n_{i,2} = n_i$). Using an argument similar to that used for Equation 2, the probability that a request that arrived in the current window, stays till the end of the current window, is given by

$$f_j(G, D) = \int_{s=0}^{D} (1 - G(D - s)) \frac{ds}{D}. \tag{3}$$

Similarly, the probability that a request that arrived j windows before, j = 1,2 stays till the end of the current window, is given by

$$f_j(G, D) = \int_{s=0}^{D} \frac{(1 - G(D - s + jD))}{(1 - G(D - s + (j-1)D))} \frac{ds}{D}. \tag{4}$$

Hence, using techniques similar to that used in the case of exponentially distributed reneging times, we can easily determine the transition matrix of the Markov chain.

It is easy to see how the above approach may be extended to the case where $W \leq lD$ w.p. 1., for any integer $l \geq 3$. However, the size of the state space will increase at an exponential rate with respect to l; it now becomes $\{\prod_{i=1}^{m}(2\bar{k}_i)\}^l$.

In summary, we see that though certain cases of on-demand video systems can be modelled as Markov chains, in most of these cases, the complexity of the solution, increases combinatorially with the number of movies.

4 Simulation Study

We now compare performance of the proposed policies using simulation. The simulated environment models a commercial video-on-demand installation with thousands of concurrent users. Client requests are modelled as a Poisson arrival process. Let λ denote the client request rate. The value of λ is chosen such that the number of simultaneous users varies from roughly 600 to 6000 (consistent with projected demand [5]). Each movie is assumed to be 2 hours long. Hence, the value of λ is varied from 5 to 50 per minute. The frequencies for requesting various movies are assumed to be a skewed distribution. We will approximate the empirical distribution in [18] (presented in Section 2) by a Zipf distribution with the parameter 0.271. We will use this distribution throughout our simulation study unless otherwise specified. The number of different movies is assumed to be 92.

The simulation keeps track of starting and ending of each movie as well as arrival, reneging and playing of each client request. For each movie with dedicated service a new playback stream is started after a batch window interval, B. For the movies without any dedicated service, once the server capacity becomes available, a new playback stream is started if it is selected by the scheduling policy. Note that with a cold start initially all movie requests will be accepted, and given the long service time duration (2 hours), it will be followed by total rejection of all requests for some time (except for the movies with dedicated service). This pattern will be repeated several times until the system reaches a steady state as a result of randomness introduced by arrival time distribution. To avoid such cyclic behavior, and to reach steady state quickly, we initialize the system in the following way. All streams are assumed to be in use, and the residual service time for each stream is assumed to be uniformly distributed between 0 and 120 minutes. The system is then run for an initial duration, T_{init}, during which no statistics are collected. The system is further run for a simulation duration of T_{sim} during which statistics are collected. In all our experiments T_{init} was taken to be 240 minutes. The duration T_{sim} was adaptively controlled so as to obtain good accuracy for reneging probability. The simulation duration consists of multiple batch durations. Each batch duration is of fixed length, T_{batch} (60 minutes in our experiments). For each batch duration, we obtain a sample of the fraction of clients that reneged. After the first 5 batches, we compute the mean and variance of the above fractions. We repeat this after each subsequent batch until the vairance drops enough to give the desired accuracy (0.1%).

4.1 Comparison of various policies

Figures 3 through 5 compare reneging probability, waiting time of the accepted clients and unfairness, resepectively, of the MQL and FCFS policies. The request frequency for various movies is given by the Zipf distribution with the parameter 0.271. We consider two cases of the customer reneging time with exponential distribution, one with a mean of 3 and the other with a mean of 5 minutes. Figure 3 shows the overall reneging probability under these two policies as a function of available server capacity. The reneging probability is lower under the MQL policy than under the FCFS policy for both values of mean waiting time. This is due to the memory-less property of the exponential distribution. At any scheduling point, the MQL selects the movie with the largest number of waiting clients, and since the reneging probability of any waiting client is the same, MQL minimizes the overall reneging probability. Note that this may not hold if the reneging probability of an waiting client depends on the amount of the time it has been in the system.

Figure 4 compares the average waiting time before a client is served. The waiting times of the reneged clients are not included in this estimation. Under the FCFS policy, a playback stream may be used just to serve a single client request while multiple client requests for a popular movie may be waiting. Therefore, the mean waiting time over all clients is higher for this policy than for the MQL policy. However, with a larger server capacity the difference between the two policies is narrowed substantially, and the absolute value becomes very small ($< 5\%$). This will be the operating range for most systems. Note that the FCFS policy is easy to implement as it maintains very little information regarding queue length, waiting information about all clients, etc.. Therefore, from practical considerations, the FCFS may be chosen over the MQL policy. The FCFS is also the fairest policy as it treats all customers equally. Figure 5 compares the unfairness of the two policies. Recall from Section 2, the higher the value of unfairness, the higher the difference in reneging probability for different movies. For a small server capacity, the MQL policy serves primarily the requests for the popular movies, while ignoring individual requests for cold movies. Hence, MQL has greater unfairness as compared to FCFS.

Figures 6 compares the reneging probability of the above two policies for uniform (0-6 and 0-10 minutes) reneging time distributions. Note that the gap between the two policies has narrowed for all cases. However, the relative order has not changed. As a general rule, however, this will not be true if the reneging probability of a waiting customer depends strongly on the current value of waiting time. Figures 7 compares the reneging probability of the two policies, where each client waits for an minimum of W_{min} minutes before reneging. The remaining reneging time is exponentially distributed with the mean 5 minutes. For a small server capacity, the MQL policy still results in a lower reneging probability. However, with increasing server capacity the relative order of the two polices are reversed. Most clients will be served for a higher server capacity. The MQL policy pays no attention to the amount of time each individual client has been waiting in the system. The FCFS policy, in contrast, always serves the client who has been waiting in the system for the longest time.

4.2 Effect of dedicated server capacity

We now focus our attention only on the FCFS policy. A variation of the basic FCFS policy, called FCFS-n policy, that dedicates a certain amount of server capacity for the most popular movies, can make a better trade-off between the fairness and the reneging probability. Under such a dedicated policy, the requests for the n most popular movies are batched every B minutes. The value of B is chosen as W_{min} or W_{max} depending on the customer reneging behavior outlined in Section 2.2. This improves resource utilization for the requests for the popular movies without increasing their reneging probability. Figure 8 shows the effect of dedicated server capacity on the reneging probability. The reneging behaviour of the customer is assumed to follow the minimum waiting time model ($W_{min} = 2$). There is very little change in the reneging probability particularly for a large server capacity. A similar observation holds for $W_{min} = 5$.

Under the maximum waiting guarantee model, clients are influenced to wait longer and hence dedicating server capacity for the popular movies can result in a lower reneging probability for a given server capacity. Figures 9 and 10 show the effect of dedicated server capacity on reneging probability for $W_{max} = 2, 5$ minutes, respectively. In both cases, there is some improvement in the reneging probability as the number of movies with dedicated streams is increased. The improvement is larger for $W_{max} = 5$ minutes. At some point, however, dedicating server capacity for the not-so-popular movies may result in inefficient utilization of resources. The optimal number of movies with dedicated service will also depend upon the arrival rate. This is further explored in the next subsection.

4.3 Capacity planning for 95% acceptance rate

In this subsection, we explore the amount of server capacity required such that the reneging probability of the clients is less than 5%. In the simulation, this is obtained by repeating the simulation following a search process on the server capacity. Since the simulation is expensive, an efficient search process is needed. For this purpose, we use a secant search method [10]. Figure 11 compares the required server capacity for MQL and FCFS to achieve the above objective (less than 5% reneging probability). The customer reneging behavior is assumed to follow the minimum waiting time model. For $W_{min} = 0$, (i.e., customer reneging time is exponentially distributed), MQL requires less capacity than FCFS. However, for $W_{min} = 2, 5$ minutes, the order of the curves is reversed. Also, the gap between the two policies increases with the arrival rate.

Figure 12 shows the effect of dedicating server capacity on the required server capacity for achieving 5% reneging probability. The customer behavior follows the maximum waiting guarantee model. (As seen before, dedicating server capacity does not make any difference to the reneging probability under the minimum waiting time model.) For the sake of comparison, we have also plotted the case of $W_{max} = 0$ i.e. where customer behavior cannot be influenced through dedication. The dashed lines represent the cases for the FCFS-20 policy (server capacity is dedicated for the top 20 movies) while the dotted lines are for FCFS-10. For $W_{max} = 2$ minutes, the curves for FCFS-10 and FCFS-20 intersect. At lower arrival rates, the FCFS-10 policy requires less server capacity than the FCFS-20 policy with the situation being reversed at higher arrival rates. This shows that the optimum number of movies with dedicated service increases with the arrival rate. A similar result can be seen for $W_{max} = 5$ minutes, except that the crossover occurs at a lower arrival rate. This implies that as W_{max} increases, increasing the number of movies with dedicated services reduces the required server capacity. Figure 13 shows the variation of required server capacity with the number of movies with dedicated service. For $W_{max} = 2$ and $\lambda = 25, 50$ the optimum number of movies with dedicated service is at 5 and 15 movies, respectively. Thus with increasing arrival rate, it is effective to have a larger number of movies with dedicated service. The corresponding curves for $W_{max} = 5$ are much flatter and hence the choice of the number of movies for which dedicated service is to be provided is less critical. Also, the optimum number of movies with dedicated service is larger than the corresponding number with $W_{max} = 2$.

We finally comment on the effectiveness of the merging policy in reducing the number of streams required by the video server. Without batching each request will require a separate stream. Then the lower bound on the capacity required for an arrival rate of λ and service duration of D can be estimated using Little's law as λD. Note that the actual capacity required to satisfy all requests will be higher due to statistical fluctuation in the arrival rate. To estimate the effectiveness of merging under various scheduling policies we also have to estimate the capacity required under a similar environment without merging. Hence, assuming 95% acceptance ratio the lower bound on the required server capacity is $0.95\lambda D$. Figure 14 shows the reduction in required server capacity due to merging corresponding to the cases in Figure 13. The reduction in server capacity is expressed as a percentage of the required capacity under no merging. Our estimate on the reduction is conservative, since we use a lower bound, rather than the actual server capacity. As can be seen from the figure, the amount of reduction depends not only on the scheduling policy (e.g, number of dedicated movies) but also on the request arrival rate. The reduction increases with the arrival rate and for $\lambda = 50$ the reduction ratio reaches 70% in this example. Therefore, batching is more effective for larger servers since there are more available requests for batching. Not all applications would demand VCR control and in a mixed environment diffeerent pricing structure can be introduced to exploit the advantage of merging.

5 Summary and Conclusions

In an on-demand video server environment, multiple clients request for playback of movies stored at a set of centralized servers. Due to stringent response time requirements, each server can play only a fixed number of concurrent streams. By batching multiple client requests for the same movie, a larger number of clients can be served for a given server capacity. In general, batching is more effective in larger servers since there are more available requests for batching. In this paper, we proposed and evaluated various scheduling policies that differ in their choice of a movie to be played when the server capacity becomes available. Merging of multiple requests into one is a novel queueing theoretic property. Analytic modeling of the proposed policies are inherently complex, and hence, the policies are evaluated using simulation. The various policies trade off various performance objectives. A longer delay in serving a client may result in reneging. Hence, an important objective of these policies is to reduce the reneging probability for all movie requests. Another objective is to reduce the average waiting time before a client is served. The two objectives may not always go hand in hand. A third objective is to be fair to all requests (irrespective of the popularity of a movie).

A FCFS policy always serves the longest waiting client, thereby ensuring fairness. It is also easy to implement in a real system as it needs to maintain very little information about individual clients. The MQL policy on the other hand serves the requests for the movie that has the largest number of waiting clients. It therefore, attempts to maximize the number of clients served, however, at the expense of fairness. If the reneging probability of a client depends on the amount of waiting, then the MQL policy may not even perform as well as the FCFS policy, since the MQL policy does not take into account the client waiting time. Therefore, FCFS is the preferred policy.

Client waiting time behavior can also be influenced to improve the performance of a scheduling policy. If the clients agree to wait for a minimum amount of time before reneging, then such information can be used to batch requests for the popular movies. One way to exploit this information is to dedicate enough capacity for the popular movies so as to start a new stream for these movies once in every batching window. Results show that such dedication is not very significant in improving the performance of the FCFS policy. A client may not agree to such minimum waiting time under uncertainty.

However, it may be willing to do so under a maximum waiting time limit. In this case, dedicating enough capacity for the popular movies may result in significant performance improvement. The optimal number of movies for which the server capacity is dedicated will depend on the maximum waiting time guarantee and the movie request rate.

In this paper, we have focussed on the scheduling policies for an environment where the application does not demand VCR like control (pause, resume, etc.). We are currently working on a follow up work that accommodates such VCR like control.

Acknowlegement: We would like to thank Bill Tetzlaff, Martin Kienzle and Steve Lavenberg for their helpful comments. We would also like to thank F. Stein for helping us in obtaining the empirical data on video rentals used in this paper.

References

[1] Anderson, D. P., "Metascheduling for Continuous Media," *ACM Transactions on Computer Systems*, Vol. 11, No. 3, August 1993, pp. 226–252.

[2] Craig, D. W., and C. M. Woodside, "The Rejection Rate for Tasks with Random Arrivals, Deadlines, and Preemptive Scheduling," *IEEE Transactions on Software Engineering*, Vol. 16, No. 10, October 1990, pp. 1198–1208.

[3] Dan, A., and D. Sitaram, "Buffer Management Policy for an On-Demand Video Server", *IBM Research Report, RC 19347*, Yorktown Heights, NY, 1993.

[4] Dykeman, H. D., M. H. Ammar, and J. W. Wong, "Scheduling Algorithms for Videotex Systems under Broadcast Delivery", *Proc ICC'86*, 1986, pp. 1847–1851.

[5] *Electronic Engineering Times*, March 15, 1993, pp 72.

[6] Fox, E. A., "The Coming Revolution in Interactive Digital Video," *Communication of the ACM*, Vol. 7, July 1989, pp. 794–801.

[7] Gross, D., and C. M. Harris, *Fundamentals of Queueing Theory*, John Wiley and Sons - New York, Chichester, Brisbane, Toronto, Singapore, 1985, pp 124.

[8] Kleinrock, L., *Queueing Systems, Volume 1: Theory*, John Wiley and Sons - New York, Chichester, Brisbane, Toronto, 1975, pp 105.

[9] Le Boudec, J.-Y., "The Asynchronous Transfer Mode: A Tutorial," *Computer Networks and ISDN Systems*, Vol. 24, 1992, pp. 279–309.

[10] Lindfield, G. R., and J. E. T. Penny, *Microcomputers in Numerical Analysis*, John Wiley and Sons - New York, Chichester, Brisbane, Toronto, 1989, pp 31.

[11] Liu, C. L., and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a hard Real-Time Environment," *Journal of the ACM*, Vol. 20, No. 1, January 1973, pp. 46–61.

[12] Marchok, D. J., C. Rohrs, and M. R. Schafer, "Multicasting in a Growable Packet (ATM) Switch," *IEEE INFOCOM*, 1991, pp. 850–858.

[13] Peha, J. M., and F. A. Tobagi, "Evaluating Scheduling Algorithms for Traffic with Heterogeneous Performance Objectives," *IEEE GLOBECOM*, 1990, pp. 21–27.

[14] Rangan, P. V., H. M. Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service," *IEEE Communication Magazine*, Vol. 30, July 1992, pp. 56–65.

[15] Sincoskie, W. D., "System Architecture for a Large Scale Video On Demand Service," *Computer Networks and ISDN System*, Vol. 22, 1991, pp. 155–162.

[16] Stankovic, J., K. Ramamritham, and S. Chang, "Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems," *IEEE Transactions on Computers*, Vol. C-34, No. 12, December 1985, pp. 1130–1143.

[17] Towsley, D., and S. S. Panwar, "On the Optimality of Minimum Laxity and Earliest Deadline Scheduling for Real-time Multiprocessors," *Proc. of EUROMICRO Workshop on Real Time*, 1990, pp. 17–24.

[18] *Video Store Magazine*, Dec. 13, 1992.

[19] Wolff, R.W., Poisson Arrivals See Time Averages, *Operations Research* **30**, **2**, 1989, pp. 223-231.

[20] Wong, J. W. and M. H. Ammar, "Analysis of Broadcast Delivery in a Videotex System", *IEEE Transactions on Communications*, Vol. 34, No. 9, September, 1985, pp. 863–866.

[21] Zhao, Z. X., S. S. Panwar, and D. Towsley, "Queueing Performance with Impatient Customers," *IEEE INFOCOM*, 1991, pp. 400–409.

[22] Zhong, W. D., Y. Onozato, and J. Kaniyil, "Copy Network with Shared Buffers for Large-scale Multicast ATM Switching," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 2, April 1993, pp. 157–165.

Figure 1: Multi-media video server environment

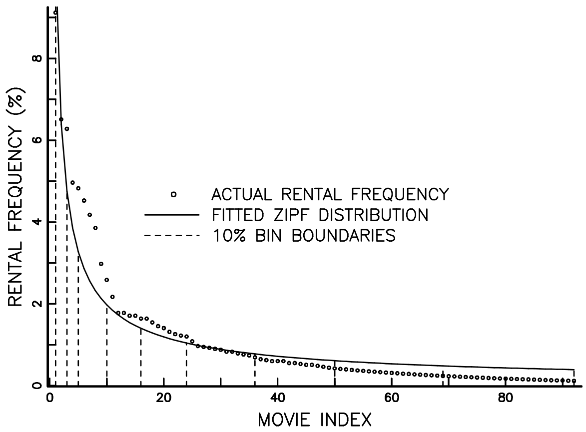

Figure 2: Rental frequency distribution

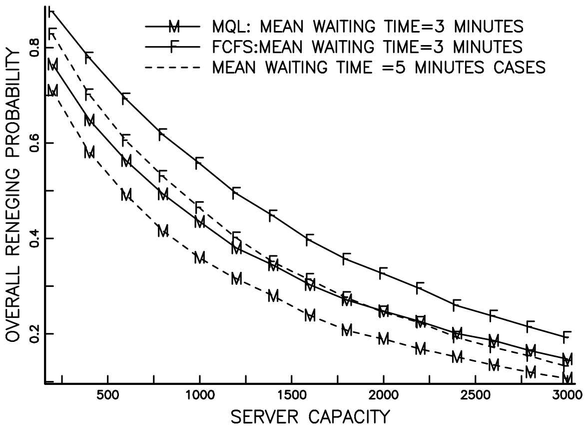

Figure 3: Comparison of reneging probability (exponential reneging time distribution)

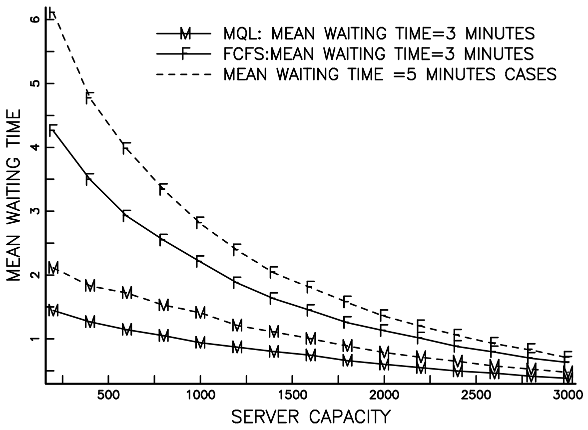

Figure 4: Comparison of waiting times of the accepted requests (exp. reneging time dist.)

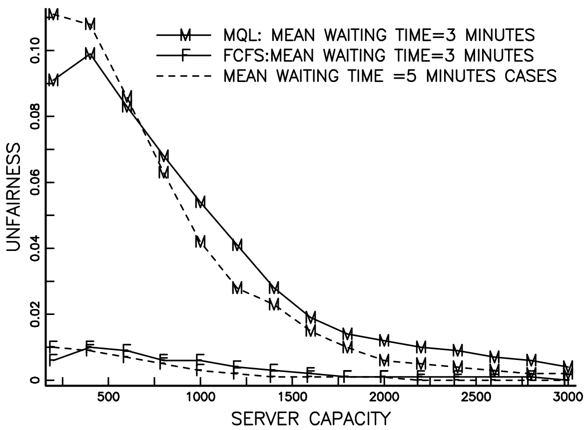

Figure 5: Comparison of unfairness (exponential reneging time distribution)

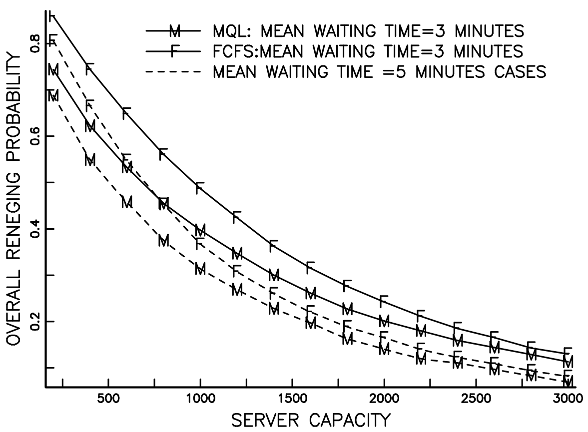

Figure 6: Comparison of reneging probability (uniform reneging time distribution)

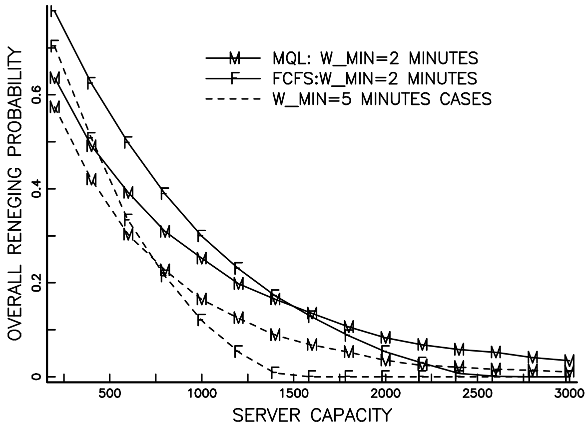

Figure 7: Effect of minimum reneging time on the reneging probability

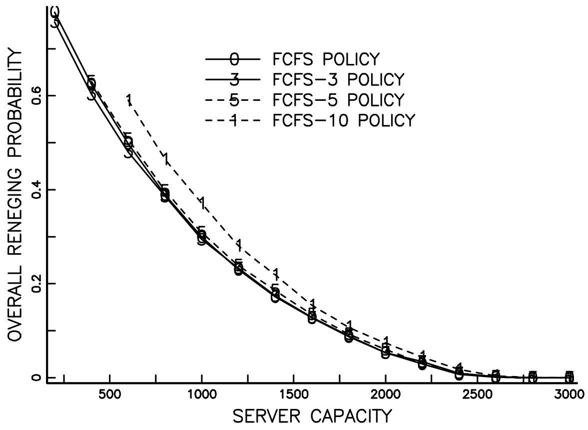

Figure 8: Effect of dedicated server capacity for hot movies on the reneging probability (W_{min} = 2 minutes)

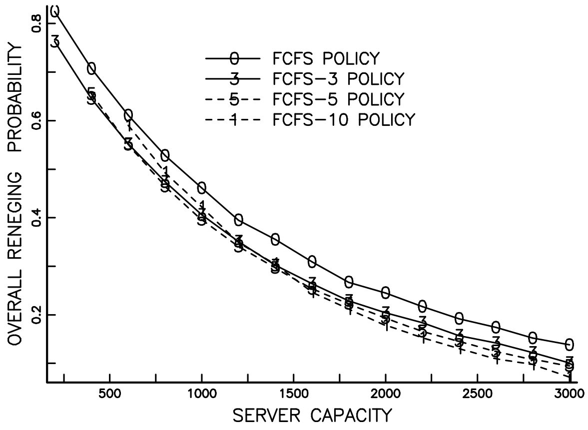

Figure 9: Effect of dedicated server capacity for hot movies on the reneging probability (W_{max} = 2 minutes)

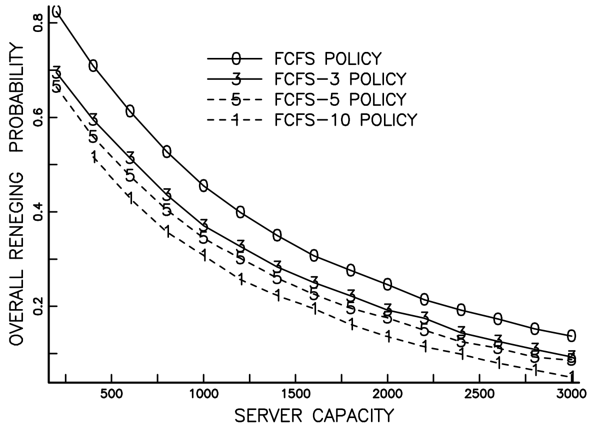

Figure 10: Effect of dedicated server capacity for hot movies on the reneging probability (W_{max} = 5 minutes)

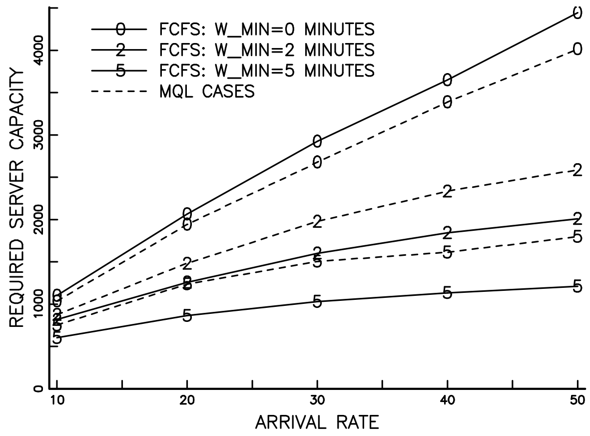

Figure 11: Effect of minimum reneging time on the required server capacity for 95% acceptance guarantee

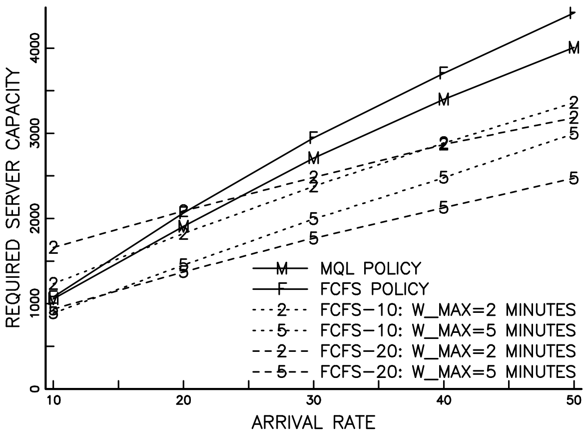

Figure 12: Effect of reneging time guarantee on the required server capacity for 95% acceptance guarantee

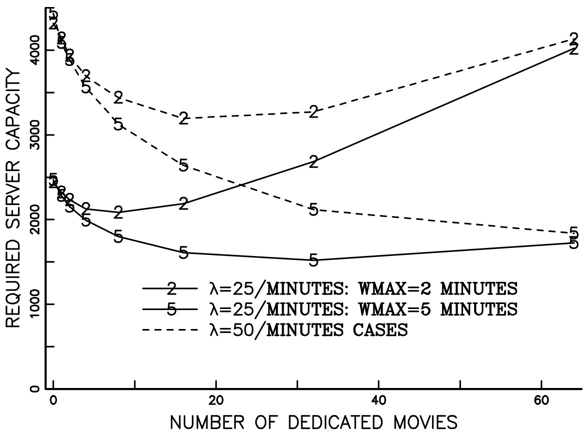

Figure 13: Effect of dedicated server capacity on the required server capacity for 95% acceptance guarantee

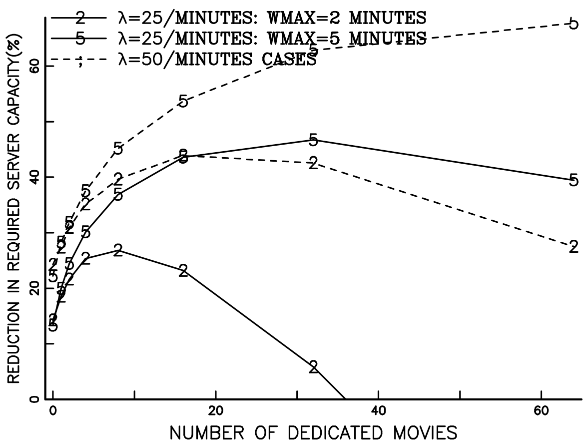

Figure 14: Effect of merging on the reduction of required server capacity