Scheduling Policies for an On-Demand Video Server with Batching

Asit Dan, Dinkar Sitaram and Perwez Shahabuddin
IBM Research Division, T. J. Watson Research Center
Yorktown Heights, NY 10598
{asit, sitaram, perwez}@watson.ibm.com

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 behavior 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 (of the order of 60%) in required server capacity can be achieved by batching.

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) [8, 15, 14]. Due to stringent response time requirements, continuous delivery of a stream has to be guaranteed by reserving the resources needed for delivery (e.g. disk bandwidth, CPU) [1, 14]. These resources are referred to as a logical *channel* in the paper. Hence there is a hard limit on the number of streams that can be simultaneously delivered by a server.

The playback requests for movies from different clients are independent of each other and may arrive at random time intervals. A new movie stream could be started to satisfy each request. Commercial environments may contain a large number of active clients, potentially in the thousands [17]. This would require a very large server capacity. However, modern communication networks such as ATM are equipped with a multicast facility [12, 11] 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, requests by multiple clients for the same movie arriving within a short time duration can be batched together and serviced using a single stream. This is referred to as *batching* in this paper. In the following, we explore various scheduling policies that arise as a result of this property.

Clients may also request pausing and restarting of movies. Such requests can be handled by setting aside a small number of channels (called *contingency channels*). This is outlined in Section 2.3 and described in detail in [4, 5].

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 their requests (or *renege*). Thus there is a trade-off between decreasing server capacity and increasing both waiting time and reneging probability. The average client waiting time is also determined by the choice of a movie to be played at any given time. The policy that chooses the movie with the largest number of waiting clients attempts to minimize the

Multimedia 94- 10/94 San Francisco, CA, USA
© 1994 ACM 0-89791-686-794/0010..$3.50

number of lost customers. However, this policy may not be fair since it does not pick the client who has been waiting the longest. 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 uncertainty. However, the client may be willing to wait if a maximum waiting time is guaranteed. This can be achieved by starting movies at predetermined intervals. Such a guarantee may be appropriate only for frequently requested movies. These interactions between client behavior 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 batching property. The non-work conserving batching 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 [19, 7]. In such a system, users submit independent requests to retrieve a specific page of information to a service computer. However, all the retrieved pages are 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 instantaneous.) 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.

A different model of batching in the context of video-on-demand systems with impatient customers was studied in [9](see also references therein). In this case it was assumed that a pre-specified number of channels is available for each video. Hence there is no competition for the channels among different movies. As a result, the scheduling policies studied in [9] are *when* to play a given video for a given number of channels for that video rather than *which* video to play on any available channel. The notion of unfairness was also not considered in allocation of channels to different videos.

Loss systems have also been studied both in the classical literature [10] and also in various communication systems [13, 20]. 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 [20] or due to missing hard real-time deadlines [16]. In our system, the individual client deadlines are not known, and the loss is primarily due to reneging of impatient clients. Also, in addition to minimizing reneging probability, we 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 trade-offs associated with these policies. Due to the batching property, the policies are hard to analyze analytically. Hence the trade-offs amongst various policies are studied using simulation in section 3. Section 4 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 [17]. It is interesting to note that the access frequencies to various movies can be characterized by a Zipf distribution with parameter 0.271.[1] 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 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% or 99%) 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 may be 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 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.

[1] In a Zipf distribution, 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.

[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.

- **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

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

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.[3] 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 [7]). Such policies are more complex and difficult to implement and will not be considered 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,[4] 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 *batching* 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 [7]. 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 movies. For all the movies with dedicated streams, a new stream is started every B minutes (referred to as *batch interval*[5]). 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 favors the hottest movies.

[3] Here we assume any movie can be played on an available resource. For example, by striping across all disks, available disk bandwidth can be used for playing any movie. For multiple striping groups, load balancing can be used as in [6].

[4] 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 allow batching.

[5] In the current paper, it is assumed that all popular movies are batched with the same interval. An analytical method for determining an optimal batching interval for each movie to minimize reneging is described in [5].

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 [16]. In general, the reneging time of a client, which we denote by R, 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 R 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 will always perform 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 reneging 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 Reneging Time:** Under this model, each customer is willing to wait at least R_{min} amount of time. The remaining reneging time is assumed to be exponentially distributed (i.e., $W = R_{min} + E$, where R_{min} is a constant and E is a exponentially distributed random variable. Hence, R_{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 Time Guarantee:** Under this model, a customer does not agree to a minimum reneging 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. The remaining customers still follow the minimum reneging time model.

2.3 VCR control

The policies proposed in Section 2.1 determine which of the requests for new movies should be scheduled. However, users may also be allowed to pause and then restart at arbitrary times. It is likely that restarting users will be willing to wait less time than users who are waiting for a movie to start.

An efficient method for dealing with pause and resume requests by setting aside a small pool of channels (referred to as *contingency channels*) is proposed in [4]. It provides a *statistical guarantee* that with high probability (e.g. 99%) a restart request will be started within a pre-specified small delay. First, if the user of a multicast stream pauses, a small amount of buffer can be used for caching the blocks of the multicast stream similar to [2]. If the pause is short, the user can be restarted without any delay. If the pause is longer, the request can be batched with an existing stream if a sufficiently close stream exists. Otherwise, the user is restarted using a free channel from the contingency pool. In the case of a user viewing a single stream, the stream is freed when the user pauses, and the channel is returned either to the contingency pool or is used for servicing a new request. A channel from the contingency pool is allocated upon restart. Note that since the method provides a statistical guarantee on the waiting time, it is not necessary to reserve a channel for each paused stream. This is more efficient than providing a deterministic guarantee, which would require channel reservation for paused streams. This method is studied in [5] where an algorithm for determining the number of contingency channels needed is described.

3 Simulation Study

We now compare performance of the proposed policies using simulation.[6] The simulated environment models a commercial video-on-demand installation with thousands of concurrent users. Client requests are modeled 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 [17]). 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 unless otherwise specified. The number of different movies is assumed to be 92.

The simulation keeps track of starting and ending of each

[6] In [3] we show that even with a lot of simplifying assumptions, both the policies are very hard to model analytically.

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_w. Movies without dedicated service are started once a server channel becomes available and the movie 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, the system is initialized as follows. 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 subdurations. Each subduration is of fixed length, T_{subdur} (60 minutes in our experiments). The simulation is run until the variance of the reneging probability (recomputed after each subduration) drops enough to give the desired accuracy (0.1%).

3.1 Comparison of various policies

Figures 3 through 5 compare reneging probability, waiting time of the accepted clients and unfairness, respectively, of the MQL and FCFS policies. 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 are studied. 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 reneging 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 state information. 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.

We have also compared the reneging probability of the above two policies for uniform (0-6 and 0-10 minutes) reneging time distributions in [3]. The gap between the two policies had narrowed for all cases. However, the relative order had 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. Figure 6 compares the reneging probability of the two policies, where each client waits for an minimum of R_{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.

3.2 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 R_{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 7 shows the effect of dedicated server capacity on the reneging probability under the minimum reneging time model ($R_{min} = 2$). There is very little

change in the reneging probability particularly for a large server capacity. A similar observation holds for $R_{min} = 5$.

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. Figure 8 shows the effect of dedicated server capacity on reneging probability for $W_{max} = 5$ minutes. There is some improvement in the reneging probability as the number of movies with dedicated streams is increased. The improvement is smaller for $W_{max} = 2$ minutes [3]. 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.

3.3 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%. The required capacity is obtained by repeating the simulation following a secant search process on the server capacity. Figure 9 compares the required server capacity for MQL and FCFS to achieve the target reneging probability. The customer reneging behavior is assumed to follow the minimum reneging time model. For $R_{min} = 0$, (i.e., customer reneging time is exponentially distributed), MQL requires less capacity than FCFS. However, for $R_{min} = 2, 5$ minutes, the order of the curves is reversed. Also, the gap between the two policies increases with the arrival rate.

Figure 10 shows the effect of dedicating server capacity on the required server capacity for achieving 5% reneging probability under the maximum waiting guarantee model. (As seen before, dedicating server capacity does not make any difference to the reneging probability under the minimum reneging 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 solid lines represent the cases for the FCFS-20 policy (server capacity is dedicated for the top 20 movies) while the dashed 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 11 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 batching 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. To estimate the effectiveness of batching under various scheduling policies we also have to estimate the capacity required under a similar environment without batching. Hence, assuming 95% acceptance ratio the lower bound on the required server capacity is $0.95\lambda D$. Note that this is a conservative estimate since the actual capacity required to satisfy all requests will be higher due to statistical fluctuation in the arrival rate. Figure 12 shows the reduction in required server capacity due to batching corresponding to the cases in Figure 11. The reduction in server capacity is expressed as a percentage of the required capacity under no batching. Our estimate on the reduction is conservative, since we use a lower bound, rather than the actual server capacity (as explained above). 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 different pricing structure can be introduced to exploit the advantage of batching.

4 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. Analytical 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 reneging 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 handling requests to start a movie. An efficient method for allowing VCR control (pause, resume, etc.) requests is detailed in [4, 5].

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

[3] Dan, A., D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching", *IBM Research Report, RC 19381*, Yorktown Heights, NY, 1993.

[4] Dan, A., D. Sitaram and P. Shahabuddin, "Scheduling Policies with Grouping for providing VCR Control Functions in a Multi-media Server", *U.S. Docket No YO993-030*, 1994 (patent pending).

[5] Dan, A., P. Shahabuddin, D. Sitaram and D. Towsley, "Channel Allocation under Batching and VCR Control in Movie-On-Demand Servers", *IBM Research Report RC19588*, Yorktown Heights, NY, 1994.

[6] Dan, A., M. Kienzle and D. Sitaram, "Dynamic Segment Replication Policy for Load-Balancing in Video-on-Demand Servers", *IBM Research Report, RC 19589*, Yorktown Heights, NY, 1994.

[7] 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.

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

[9] Gelman, A. D. and S. Halfin, "Analysis of Resource Sharing in Information Providing Services," *Proc. IEEE Global Telecommunications Conference and Exhibition 1990*, Vol. 1, 1990.

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

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

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

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

[19] 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.

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

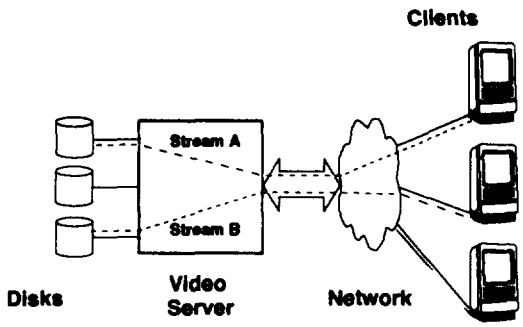

Figure 1: Multi-media video server environment

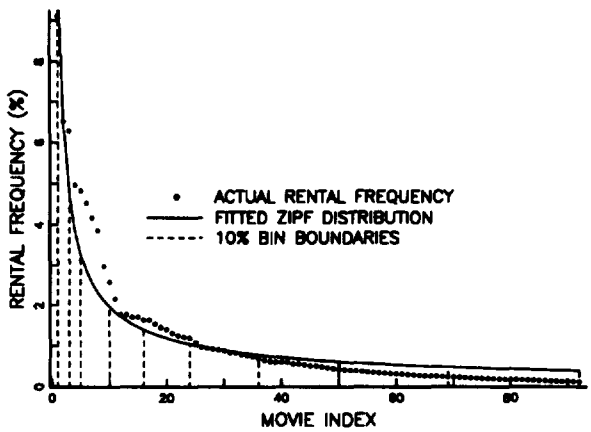

Figure 2: Rental frequency distribution

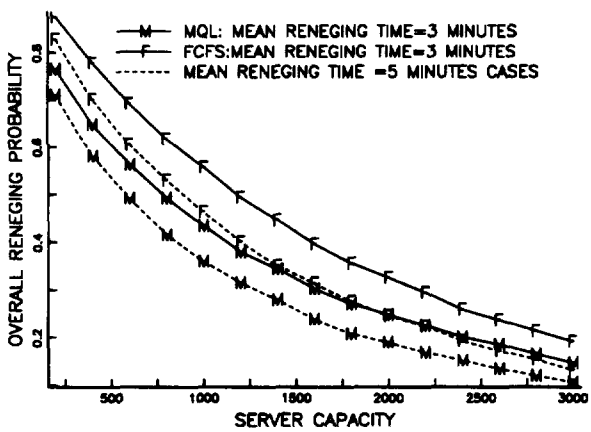

Figure 3: Comparison of reneging probability (exponential reneging time distribution)

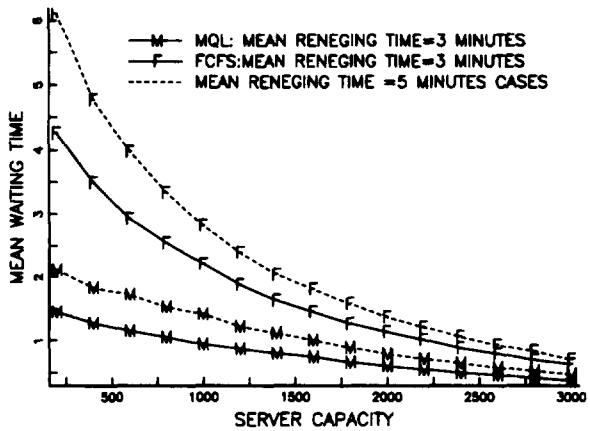

Figure 4: Comparison of waiting times of the accepted requests (exponential reneging time distribution)

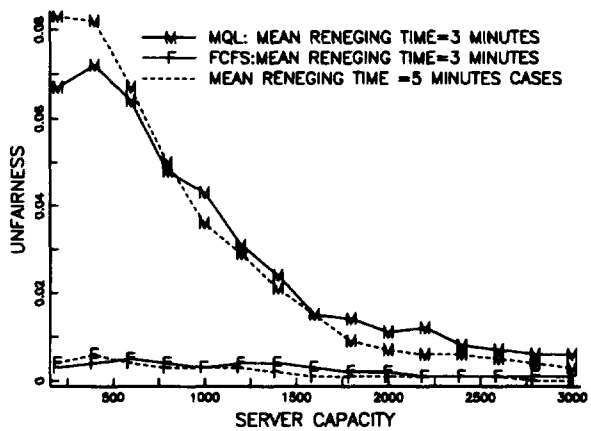

Figure 5: Comparison of unfairness (exponential reneging time distribution)

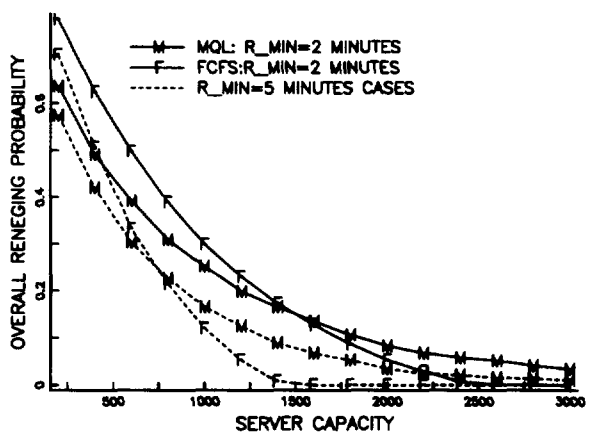

Figure 6: Effect of minimum reneging time on reneging probability

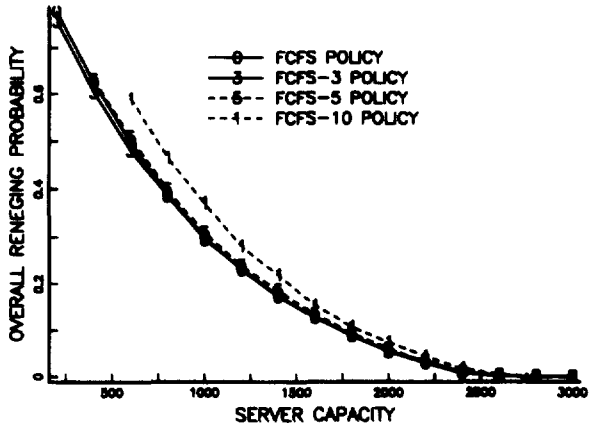

Figure 7: Dedicated server capacity for hot movies (R_{min} = 2 minutes)

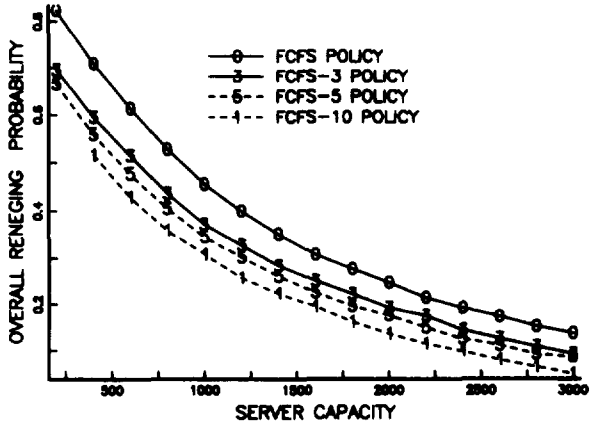

Figure 8: Dedicated capacity for hot movies (W_{max} = 5 minutes)

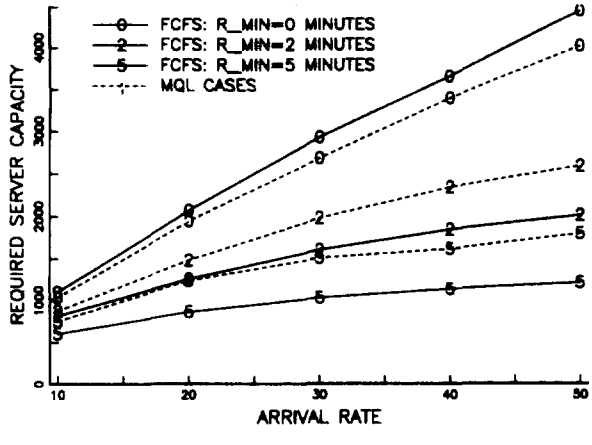

Figure 9: Effect of minimum reneging time on required server capacity for 95% accept. guarantee

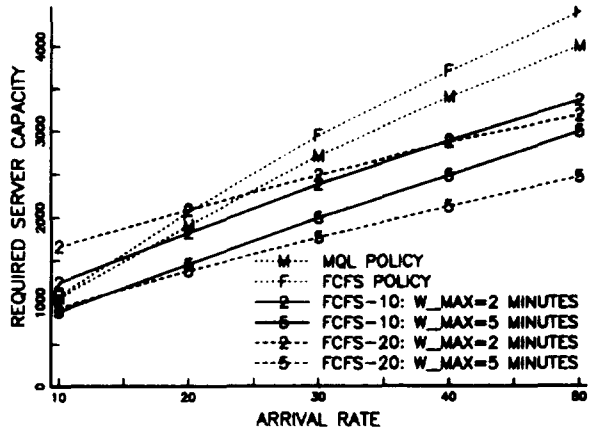

Figure 10: Effect of maximum reneging time guarantee on required server capacity

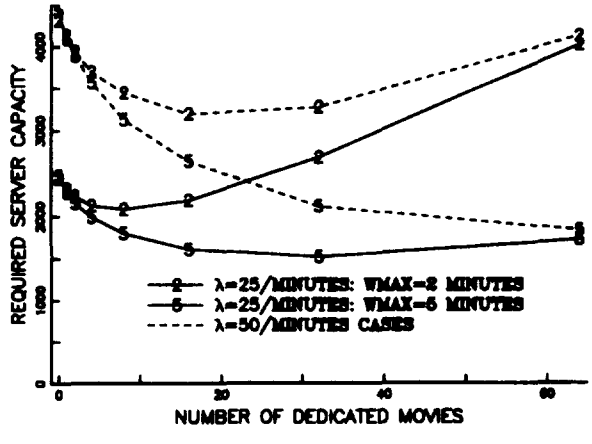

Figure 11: Effect of dedicated server capacity for the hot movies

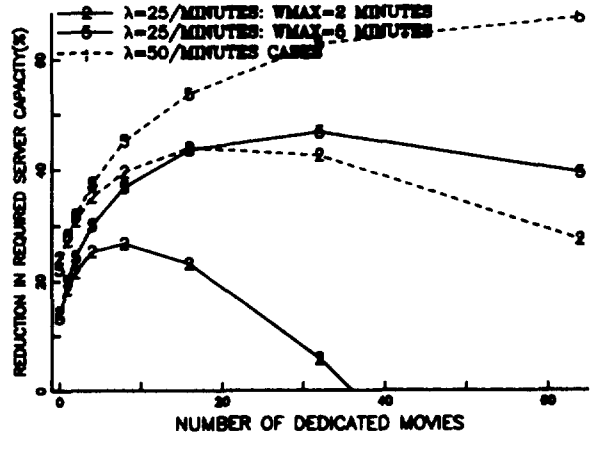

Figure 12: Effect of batching on the reduction of required server capacity