Long Term Resource Allocation in Video Delivery Systems

Kevin C. Almeroth (contact author)
Networking and Telecommunications Group
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
kevin@cc.gatech.edu
(404)894-9911(office)
(404)894-0272(FAX)

Asit Dan
Dinkar Sitaram
William H. Tetzlaff
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{asit,sitaram,tetzlaf}@watson.ibm.com

Georgia Tech Technical Report: GIT-CC-95-45
IBM Research Report: RC 20249

September 1, 1995
Revised: June 14th, 1996

Abstract

In typical video delivery systems offering programs on-demand, service should be be nearly immediate and continuous. A video server can provide this type of service by reserving sufficient network and server resources for the duration of playout. Scalability and reduced cost can be achieved using a single channel to serve multiple customers waiting for the same program (referred to as *batching*). Batching is especially useful during high load periods typically occuring during evening prime time hours. Typical channel allocation algorithms use a greedy, allocate-as-needed policy. Variations in system load can cause these algorithms to suffer poor and unpredictable short-term performance, and non-optimal long term performance. In this paper, we develop a set of realistic workloads, identify the limitations of greedy allocation algorithms, and propose a set of rate-based allocation schemes to solve these limitations. The performance of various video delivery systems are simulated and compared. The rate-based policies are shown to be robust for the workloads examined, and are easy to implement.

1 Introduction

One paradigm for the delivery of video-based services is to stream video data on an as-needed basis from the server to one or more customers. The scheduling and delivery of video is made in response to program requests sent to the server by customers. In such systems, server and network resources (together referred to as a *logical channel*) are reserved for the duration of the program. One logical channel is sufficient to meet the real-time bandwidth demands and continuous delivery requirements of one video stream[1, 2, 3, 4].

Because resources in both the server and network are finite, there is a hard limit on the number of channels that can be simultaneously delivered[5, 6, 7, 8]. However, in the presence of a multicast facility, multiple customers requesting the same program within a short interval of time can be served by the same video stream. This policy, referred to as *batching*, takes advantages of high request rates among popular programs and can result in substantial savings in server and network capacity [9, 10, 11, 12, 13, 14].

One of the more difficult tasks in a video delivery system is scheduling the finite number of channels in a way that optimizing some set of performance measures. One of the most important criteria is to make the most money possible which loosely translates into maximizing the number of customers serviced over the long term. One common approach to achieving this objective is make customers wait as long as possible in the hopes that additional requests for the same program will be made. However, these algorithms have several significant drawbacks: (1) they do not minimize customer waiting time, (2) they perform poorly when load conditions vary meaning they do not even achieve their goal of maximizing customers serviced, and (3) they have wild oscillations in short-term performance. As we will show, the fundamental flaw in these algorithms is that they use the greedy approach and do not control the rate of channel allocation. This is especially damaging considering that once a channel is allocated it is held for the duration of the program which can be a couple of hours long. If a large percentage of channels are allocated over a short period of time, the system will have few channels with which to satisfy additional requests. The system will be forced to reject a majority of requests until the first set of channels become available. As we will show, better performance can be achieved by using an algorithm which controls the channel allocation rate.

This paper documents our efforts in three areas. First, we develop a series of workload models that are not steady state Poisson models and more accurately reflect customer behavior over a 24-hour period. Second, using a video delivery system model, we simulate the performance and show the limitations of several greedy channel allocation algorithms. Third, we propose a set of rate-based channel allocation algorithms that solve the limitations of the greedy algorithms and show this by comparing simulation results.

The remainder of the paper is as follows. In Section 2, we describe the architecture of a video delivery system. Section 3 offers system and customer behavior models which are used as a basis for simulating system operation and measuring performance. In Section 4, we study the performance of two greedy channel allocation policies and identify their limitations. Subsequently, in Section 5, we propose a set of rate-based allocation policies to solve these limitations. Section 6 presents additional results on the performance improvements gained by using rate-based algorithms. The paper is concluded in Section 7.

2 Architecture

Video delivery systems can vary in size and complexity depending on many factors including the number of simultaneous streams the system is capable of providing, and the type of service to be offered. Systems can have servers ranging from one small, centralized machine to a large network of distributed file servers with several control points. The network can also varying significantly; most importantly in what transport layer functions are provided. Any number of protocols over a variety of media and topologies may be available. In order to eliminate unimportant details of any one particular system, we propose the use of a generic architecture. An overview of this system with specific emphasis on the *scheduler* is described in this section.

2.1 Video System Architecture

Figure 1 shows a generic architecture with three main components: video server, network, and set of clients/customers. The typical operation is based on customers making program *requests* to

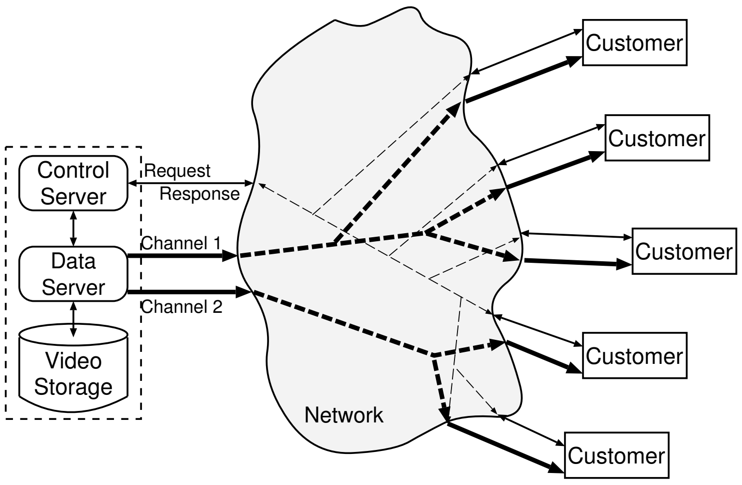

Figure 1: Generic system architecture with multicast capability.

the video server. These requests flow over a two-way, low bandwidth channel established for the purpose of exchanging control messages. A request is received and queued by the control server until the *scheduler* is ready to allocate a logical channel for the playout of the requested program. A wide variety of policies ranging from simple on-demand strategies to more sophisticated rate-control strategies can be used. We assume that the network provides efficient multicast communication which means that if multiple requests for the same request are in the queue, they can be *batched* and serviced with one logical channel.

2.2 Channel Scheduling Architecture

The process of allocating a channel for the playout of a particular program is dependent on two factors: (1) which program to play, and (2) when to start playout. Choosing the order in which to satisfy requests can be a complex decision depending on fairness issues and the number of service classes. For example, one request might receive deluxe or preferred service for a higher fee and thus be served on a priority basis. However, in this paper, we assume a first come, first served (FCFS) policy because it is fair, effective, and easy to implement[10]. In accordance with FCFS, the request in the front of the queue is served first and all other requests waiting for the same program are batched and serviced together. Program playout is decided by the channel allocation policy. The analysis of different policies is the basis of this paper and they are discussed in Sections 4 and 5.

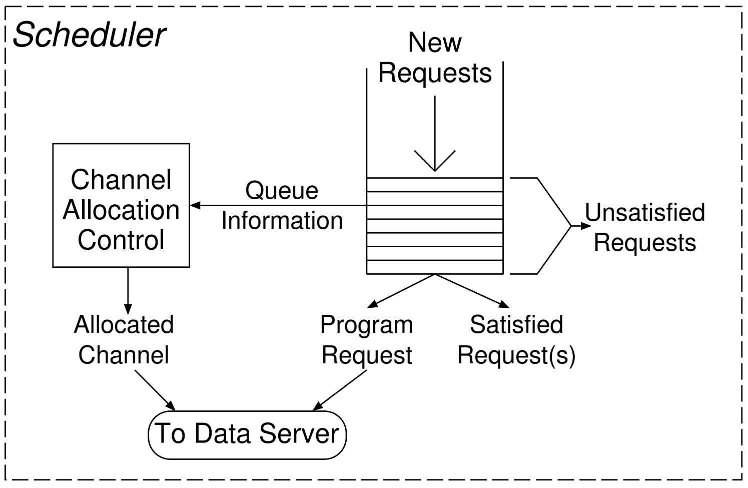

Figure 2: Outline of scheduler and channel allocation function.

2.3 Providing VCR Control

Irrespective of initial channel scheduling policies, system needs to provide means for supporting VCR control operations (e.g., pause/resume, rewind, fast forward). In general, a new channel has to be allocated when a customer initiates a VCR action and is no longer synchronized with the original batched stream. In [9, 10, 11, 15], it is proposed that some channels should be set aside to provide continuous service for such unpredictable VCR operations. These channels are referred to as *contingency* or *emergency* channels. Because the allocation of contingency channels is independent of the initial channel allocation policy we do not consider specific issues dealing with the provision of VCR operations. However, the overall affect of VCR operations is simulated by adding or subtracting a random amount of time to the expected duration of a program.

3 Video System Model

In order to compare the performance of various channel allocation policies, we must first develop a model to use as a basis for simulating the operation of various system configurations. The parameters used to describe a system's characteristics can be grouped into two categories: (1) parameters dealing with a system's size and operation, and (2) parameters dealing with customer behavior. A third group of parameters are the performance measures used to compare systems. The components of the system model are described next.

3.1 System Characteristics

1. **Logical Channel Capacity**: The number of simultaneous video streams that can be supported. The results presented in this paper are for systems with 1000 logical channels. Other numbers were tested but results are not included. Generally, as systems grow in size, additional economies of scale can be achieved. We consider 1000 channels to be a mid-sized system.

2. **Movie Library**: The number and duration of offered programs. Results are presented for systems offering 100 movies that last a uniformly random time between 105 and 135 minutes. Other variations of these numbers were tested and results suggest that a larger number of movies reduces performance by reducing the likelihood of batching.

3. **Batching Option**: Batching, if implemented by a system, allows multiple requests to be serviced with one channel. Simulations are run for both types of systems.

4. **Channel Allocation Policy**: The policies discussed in Sections 4 and 5 are each simulated.

3.2 Customer Behavior

1. **Request Arrival Pattern**: Simulations are run for a 24-hour period from 8am of one day to 8am of the next. Customer behavior defines the request arrival pattern or *workload* at the server. Over a 24-hour simulation period the workload is expected to vary depending on the time of day[16, 6]. In Figure 3, we propose three new non-steady state workloads. The horizontal axis represents time, and the vertical axis shows the number of requests that arrive during a six minute *reporting interval*.

   The first part of Figure 3 shows a steady-state request arrival pattern computed using an exponential function. This is unrealistic because demand fluctuations exist especially during "prime time" viewing hours[17]. One workload based on known viewing patterns assumes the request rate between 6pm and 12am will be five times greater than the rest of the day. The second and third graphs in Figure 3 show two such workloads; one with a gradual increase to high load, and one with a jump to high load. Variations in workload are created using

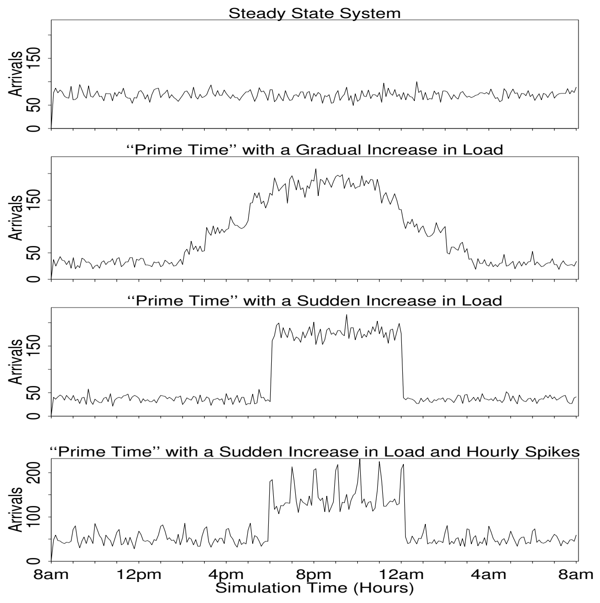

Figure 3: Various request arrival patterns.

exponential functions with different average inter-arrival times. The fourth workload is a special case with load surges at hour intervals. This type of workload is based on the belief that broadcast programs that start and end at the start/end of each hour will still exist and impact on-demand behavior.

2. **Movie Selection**: For each request, a movie is selected using a Zipf distribution[18] which states that the probability that movie i is chosen equals $\frac{c}{(i^{1+\theta})}$, where c is a normalizing constant. Empirical evidence presented in [17] suggests that $\theta = 0.271$. The Zipf distribution roughly translates to an above average number of requests for popular or *hot* movies and less frequent requests for unpopular or *cold* movies.

3. **Reneging Behavior**: Various video delivery service models have been proposed[9, 10, 6] and simulated. The model we use in this study is based on customers unwillingness to wait for service more than some finite amount of time[19]. Because servicing is first come, first served, all requests will eventually be satisfied. However, during high-load periods a customer's wait time may be very long (on the order of 15 or more minutes). Customers will most likely be unwilling to wait this long. Therefore, our model assumes that customers agree to wait at least 5 minutes plus up to an additional 2.5 minutes randomly chosen from an exponential

distribution. If after this period of time a channel has not been allocated to satisfy the customer's request, the customer withdraws the request and *reneges*. Customer reneging is analogous to a request that cannot be satisfied and is a key performance measure.

3.3 Performance Objectives

1. **Average Wait Time**: Wait time is measured from when a customer makes a request to when video playout begins. Average wait time can be measured over short intervals (6 minutes) or over the entire simulation period.

2. **Reneging Probability**: The probability of reneging is measured by taking the number of customers who renege divided by the total number of requests made. The reneging probability is also measured over the short- and long-term.

4 Greedy Allocation Policies

In this section, we describe the operation and performance of video delivery systems that implement greedy channel allocation policies. The two policies discussed are "on-demand allocation" and "forced wait". While neither policy controls the rate of allocation, both do batch requests when conditions allow. That is, when a channel is allocated, all requests in the queue waiting for the particular movie are serviced. The details of each policy and their performance are described next.

4.1 On-Demand Allocation

Allocation of channels using on-demand scheduling is the most straightforward of all the policies discussed in this paper. The algorithm is as follows:

1. When a request is received, the scheduler checks to see if there are any free channels. If a free channel exists, it is immediately allocated. Otherwise, the request is queued.

2. When a channel is freed, the scheduler checks to see if there are any requests in the queue. If requests are waiting, the freed channel is immediately allocated.

Typical operation during low-load periods is that there will be at least one free channel each time a request arrives. This means no requests will be queued, no batching will occur, and all requests are satisfied. However, during high-load periods, requests arrive faster than channels are freed so request queuing and batching are common. Because of increased queuing, wait times increase and some customers renege.

Figure 4 is the first of a series of time dependent graphs that show the performance results of simulated video system operation. The horizontal axis represents time and ranges over a 24-hour period from 8am of one day to 8am of the next. Points in the graphs represent averages over 6 minute *reporting intervals*. The dotted lines represent cumulative averages computed from the start of the simulation. Each graph has 5 parts:

1. **Arrivals**: The number of requests received by the scheduler.
2. **% Renege**: The percentage of requests reneged by customers who have waited too long.
3. **Channels Alloc**: The actual number of channels allocated during a reporting interval.
4. **Wait(secs)**: The average wait time of all successfully satisfied requests.
5. **In Queue**: The average number of requests waiting in the queue.

Figure 4 shows the performance of a video system that implements on-demand channel allocation[1].

This system has poor long-term reneging and very pronounced oscillations in short-term reneging (see second graph in Figure 4). Oscillations in short-term reneging occur because channels are allocated more quickly than they are being freed. When the load jumps at 6pm (see top graph in Figure 4), there are only enough channels to satisfy requests on an on-demand basis for a half hour (see third graph in Figure 4). After this time, all free channels have been allocated. Furthermore, since most of the channel pool was just allocated, these channels will not be available for the remainder of their service time (approximately two hours). Until these channels are eventually freed a majority of new requests will not be satisfied (see the second graph in Figure 4). When the group of channels does eventually becomes free they will again be completely allocated in a short period

[1]Results are shown only for the workload with a sudden increase in load during prime time. Results for the other non-steady state workloads discussed in Section 3.2 are almost identical and are not included.

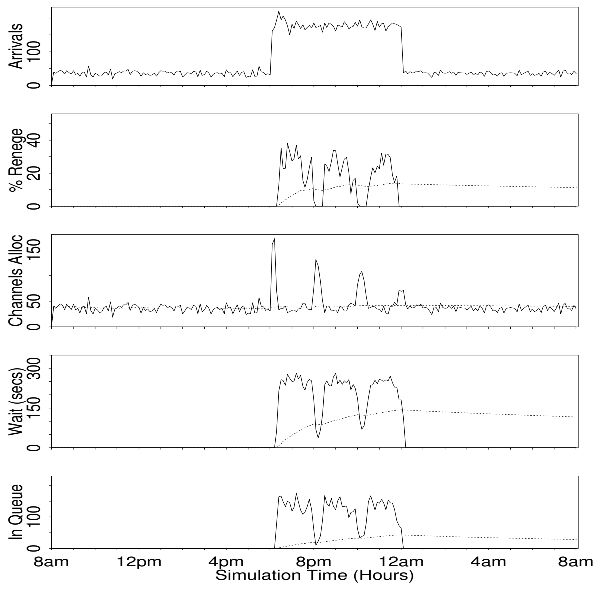

Figure 4: The on-demand channel allocation policy.

of time (see the third graph in Figure 4). These cycles eventually go away, but it takes many hours and does not occur until after the prime time period is over.

Figure 5 shows an enlargement of the time period from 6pm to 11pm. The additional detail shows more clearly the oscillation in short-term performance, and, more importantly, the duration of high reneging, and long customer wait time. Cycles occur over a two hour period because the average service time is two hours. During a cycle, the system has acceptable behavior for only 25% of the time. For the other 75% of the time, average reneging is greater than 40%, and sometimes even goes above 80%!

4.2 Forced Wait

Forced wait is a scheduling policy requiring the request at the head of the queue to wait some minimum amount of time before being serviced. The goal of the *minimum wait time* parameter is to make customers wait as long as possible thereby increasing the batching factor and effective server capacity. With a forced wait algorithm, channels are allocated on an on-demand basis but only after the request at the head of the queue has waited the minimum wait time.

Forced wait suffers from several drawbacks. The first of which is that the minimum wait time

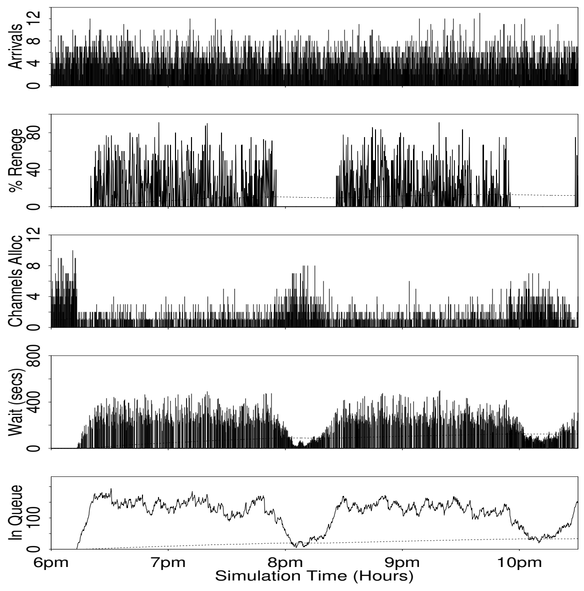

Figure 5: Enlargement of cycles caused by on-demand allocation.

parameter must be set by a system operator. Finding the optimal value is very difficult for three reasons: (1) customer behavior changes over time, (2) different values are appropriate for different workloads, and (3) system performance is very sensitive to the parameter. Figure 6 shows the reneging levels for the same video delivery system but with three different minimum wait times. For a system with 4, 5, and 6 minute minimum waiting time, the average reneging is 8.9%, 6.2%, and 17.6% respectively. The optimal minimum wait time is slightly more than 5 minutes and is a result of the customer behavior model used in the simulation. Other behavior models that do not assume all customers are willing to wait a least 5 minutes will actually perform worse. Forced wait performs especially poorly when channels are available but a customer is unwilling to wait the minimum wait time and reneges.

A second major drawback of the forced wait algorithm is that it does not fix the problem of cyclical behavior even using the optimal minimum wait time. Figure 7 shows the results are similar to the on-demand policy. When the load increases, all free channels are allocated in a short period of time, starting the cyclical behavior. One difference is that the duration of high reneging periods in forced wait is not as long as with the on-demand policy. Still, short-term reneging is unacceptable for almost half of each two hour cycle.

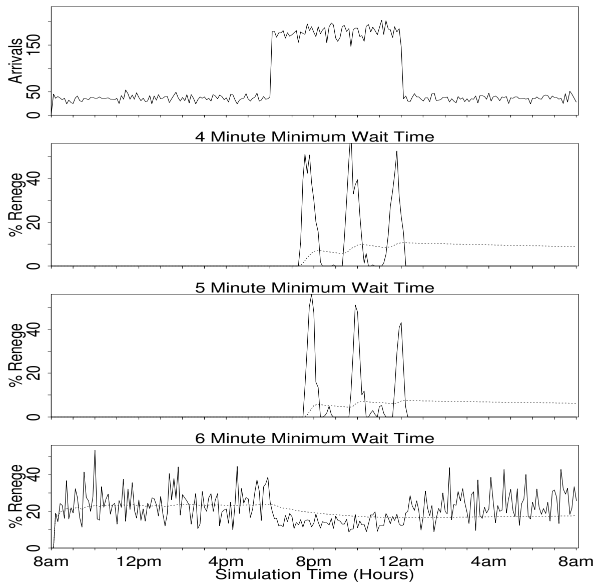

Figure 6: The forced wait policy with different minimum wait times.

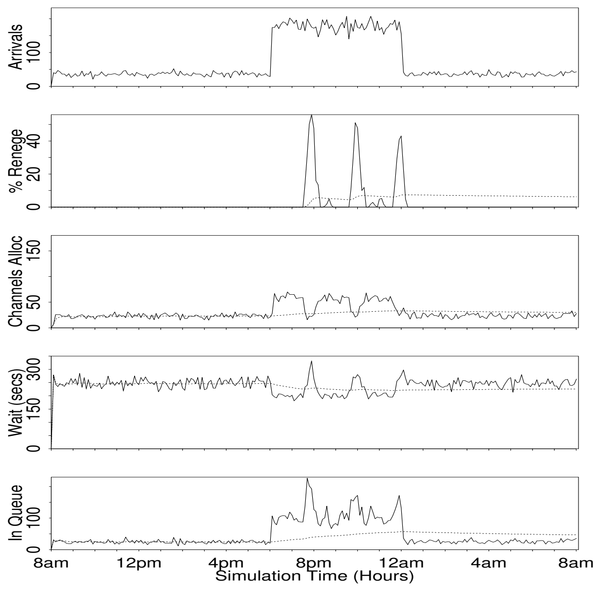

Figure 7: The forced wait policy with optimal minimum wait time.

A third major drawback of forced wait is the high waiting time experienced by each customer (see the fourth graph in Figure 7). Even during low loads, many customers are forced to wait the minimum waiting time. This policy is certainly not appropriate for systems that do not experience high load all the time. During low load periods customers should be serviced more quickly. Furthermore, for unpopular movies, the likelihood of having multiple requests in the queue at the same time is very low. Therefore, using a forced wait policy for these requests does nothing to increase effective capacity and only increases the average wait time.

5 Rate-Based Allocation Policies

Greedy allocation policies suffer from poor performance in terms of short- and long-term reneging and average wait time. In this section, we propose a set of channel allocation policies which solve these problems by controlling the rate at which channels are allocated.

5.1 Pure Rate Control

The pure rate control policy (1) puts a strict upper limit on the rate at which channels can be allocated and (2) limits the total number of channels allocated during a fixed time interval or *control interval.* The purpose of the interval is to provide a finite time period over which to accurately calculate and enforce the rate control policy. By limiting the allocation of channels to a set rate, we can ensure that some channels will be available in each control interval. This also serves to elimination the conditions leading to the development of cycles.

The pure rate control policy attempts to allocate channels at a uniform rate throughout each interval. Let T_{last} be the last time a channel was allocated, and T_{next} be the next time a channel can be allocated. T_{next} is given by $T_{next} = T_{last} + \Delta$ where Δ is the interval between successive allocations. Let T_{left} be the remaining time until the end of the control interval, C_{max} be the maximum number of channels that can be allocated during each interval, and C_{alloc} be the number of channels already allocated. Then, Δ can be computed as $\Delta = T_{left}/(C_{max} - C_{alloc})$.

The maximum number of channels that can be allocated during an interval depends on the average channel holding time (i.e., viewing time), T_{view}, and the total number of channels in the

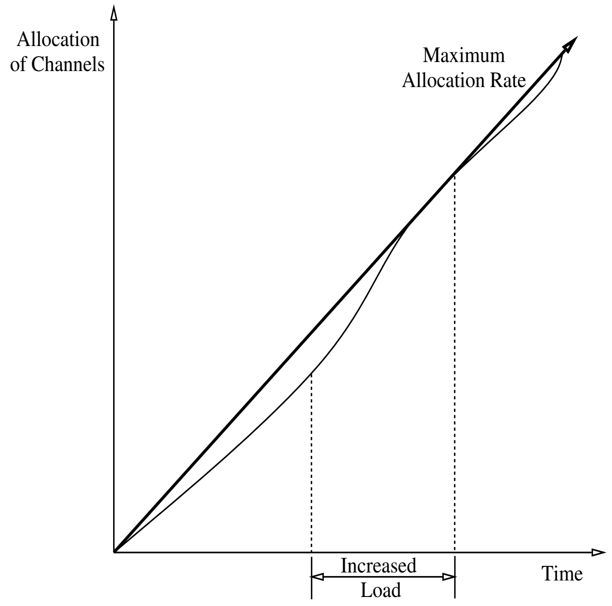

Figure 8: Maximum allocation rate for pure rate control.

system, C_{total}. Let T_{int} be the length of the control interval. Then the maximum number of channels that can be allocated per time interval, C_{max}, is computed as $C_{max} = (T_{int} * C_{total})/T_{view}$. The video server simulated in this paper is characterized by $T_{int} = 10$ minutes, $C_{total} = 1000$ channels, and $T_{view} = 2$ hours. This yields a value of $C_{max} = 84$ channels.

This result and the allocation rate formula together establish a maximum allocation rate (see Figure 8. This limit controls the channel allocation rate within a control interval. The relationship between various request arrival patterns and these equations can be summarized as the following:

- The request arrival pattern during low load periods will not use channels as fast as the allocation rate might allow. In most cases, the request inter-arrival time will exceed the delay imposed between channel allocations. During low load periods, requests will be satisfied immediately.
- When the request arrival pattern is high enough, requests will arrive more quickly than channels can be allocated. Requests will be queued and must wait for the scheduler to allocate a channel.
- When the workload transitions from low to high load, the allocation rate can be increased

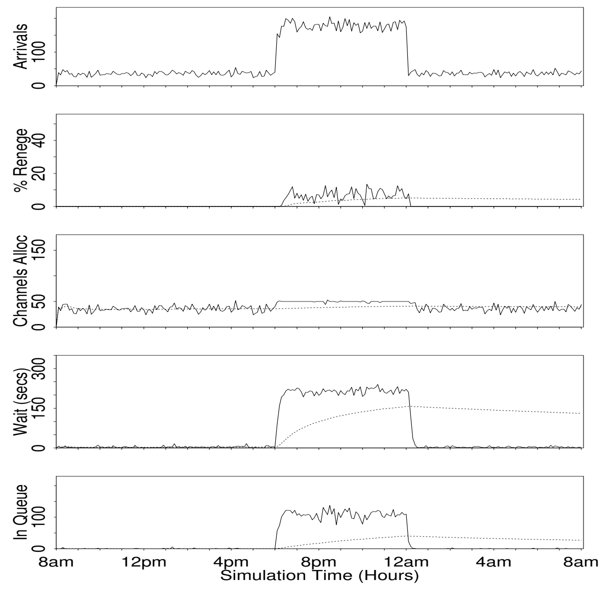

Figure 9: The pure rate control policy.

up to the maximum allocation rate. Figure 8 shows such an example. When the increased load period begins, the number of allocated channels is below the maximum. The scheduler allocates channels at an increased rate until the average equals the maximum. This mechanism allows pure rate control some flexibility in adapting to changes in workload. In the next section we will look at further relaxing this control.

Figure 9 shows the performance of a system using pure rate control. The results show that pure rate control solves all of the problems of the on-demand and forced wait policies. Cycles are eliminated because the number of channels allocated during the high load period does not spike, but only increases slightly. When the load is low the average wait time is almost zero, and there are few requests in the queue. During high load periods, channels are allocated at the maximum rate and some requests are queued. Multiple requests in the queue increases the likelihood that the system will be able to batch thereby increasing effective throughput. Rate control ensures reasonable channel availability and wait times even during high load periods.

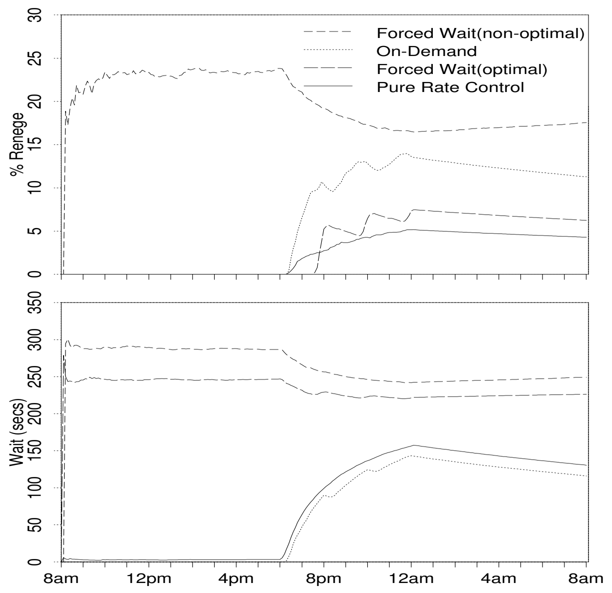

Figure 10: Comparison of channel allocation policies.

5.1.1 Advantages of Rate Control

Rate-based channel allocation combines the best characteristics of the on-demand and forced wait policies. The results in Figure 10 compare the cumulative long-term average reneging and wait times for these policies. The first part of the graph shows that pure rate control has the best overall reneging even outperforming forced wait. Furthermore, pure rate control does not require the determination of the minimum wait time parameter. The second part of the graph shows that the average wait time for pure rate control is almost optimal. Only on-demand allocation is better, but of course it suffers from both short- and long-term average reneging.

5.2 Extensions to Pure Rate Control

In this section, two extensions to pure rate control are proposed and their performance simulated. The goal of the first extension is to improve performance by relaxing the strictly enforced maximum allocation rate. The second extension looks at ways different classes of service can be supported.

5.2.1 Deviated Rate Control

Deviated rate control is similar to pure rate control, except that under certain conditions deviated rate control allows the channel allocation rate to temporarily rise above the maximum allocation rate (see Figure 11). However, the maximum number of channels that can be allocated during a control interval is still the same. Deviated rate control simply provides more flexibility in allocating channels in an interval. This additional flexibility is an important feature that can further reduce reneging during short-term load surges.

Deviated rate control requires two parameters to be defined, (1) the *maximum allowed deviation*, and (2) a *maximum tolerable wait* threshold. The maximum allowed deviation, which is defined between one and zero, shortens the delay between one channel allocation and the next. The time of the next channel allocation is still $T_{next} = T_{last} + \Delta$, but Δ becomes Δ' which is now computed as $\Delta' = \Delta * \alpha$ where α is the maximum allowed deviation.

Deviation from the ideal allocation rate should only occur when load conditions are such that either a channel be allocated quickly or customers will start reneging. This goal is achieved by allowing the allocation rate to exceed the ideal rate only when the customer at the head of the queue has exceeded the minimum wait time threshold and is believed to be close to reneging. The maximum allowed deviation parameter can be used to implement a rate-based policy with any level of control. A value of one implements the pure rate control policy while a value of zero is the same as forced wait but with a limit on the total number of channels that can be allocated per control interval.

Figure 12 compares the performance of the rate control schemes for a workload with demand spikes on each hour. This alternate type of workload demonstrates the types of scenarios in which deviated rate control performs better than pure rate control. As Figure 12 shows the long-term averaging reneging is 2.3% for pure rate control and 1.5% for deviated rate control. For the results shown, the maximum allowed deviation parameter is set to 0.45 (determined to be optimal through simulation) and the maximum tolerable wait threshold is 5 minutes. Allowing the allocation rate to deviate gives the scheduler greater ability to handle short-term surges in demand.

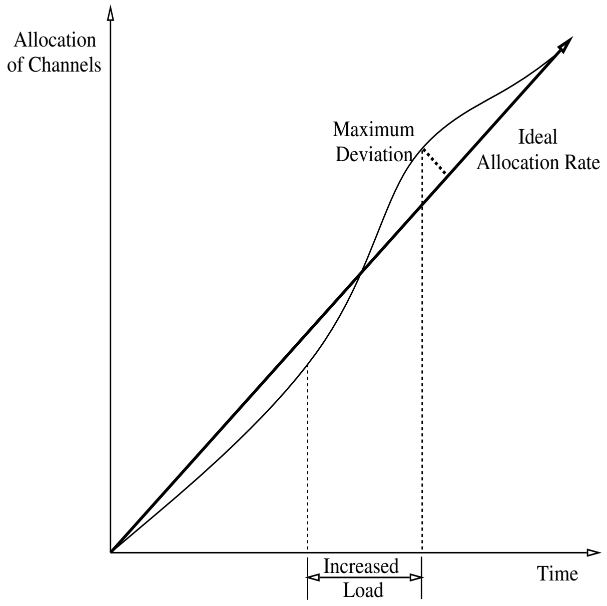

Figure 11: Maximum allocation rate and maximum deviation for deviated rate control.

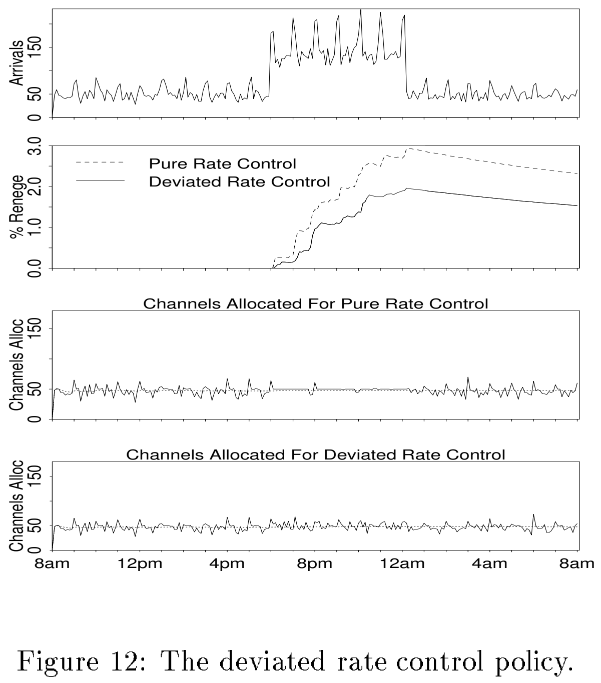

Figure 12: The deviated rate control policy.

5.2.2 Multiple Service Classes

An allocation rate control policy using multiple service classes gives a service provider the ability to offer a variety of service classes. One example is to give higher scheduling priority for hot movies since the likelihood of batching multiple requests is greater thereby increasing the profit potential. The determination of which movies are popular and which are not can be made many ways. The method used in this paper, is to use the last control interval's request count as a heuristic. Any movie with enough requests is considered hot. The allocation policy for each class is:

1. **Hot Movies**: The first request for a hot movie must wait the minimum wait time, similar to the forced wait policy. As soon as this condition is met, a channel is immediately allocated and all requests for the movie are satisfied. Hot movies have scheduling priority over cold movies, and can be guaranteed to start within the minimum wait time.

2. **Cold Movies**: Scheduling for cold movies is done using pure rate control. However, any channels allocated for hot movies are still counted in the total number of channels that can be allocated in a control interval. Starvation is possible but unlikely since only a few movies are typically hot.

Figure 13 shows the performance of the multiple service class allocation policy. Only during the high load period were there enough requests to make any of the movies hot. During the entire simulation no request for a hot movie ever went unsatisfied (see the second graph in Figure 13). Because only a few movies were hot, only a few channels needed to be allocated for these movies (see the third graph in Figure 13). Wait time for hot movies was slightly less than for cold movies (see the fourth graph in Figure 13). Finally, only a small percentage of the queue was actually for hot movie requests (see the fifth graph in Figure 13).

5.3 Comparison of Allocation Policies

In this section we compare the cumulative long-term averages of three rate-based allocation control policies and the forced wait policy with an optimal minimum wait time. Figure 14 shows that all three rate-based policies have lower long-term reneging and much better average wait times.

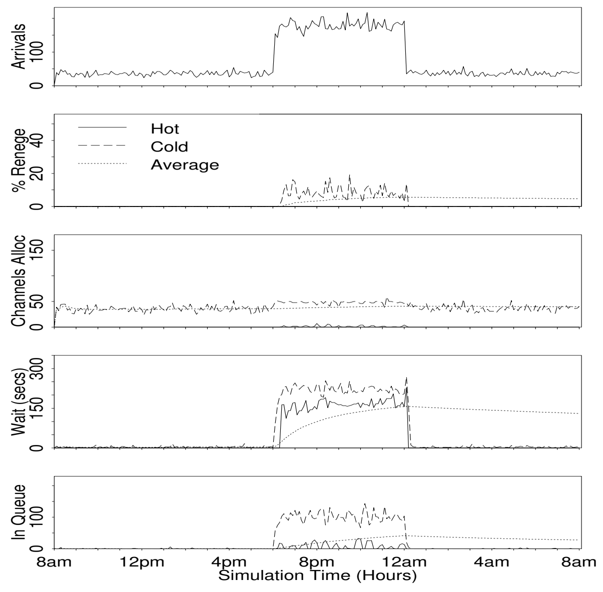

Figure 13: The multiple service class allocation policy.

Among the rate-based policies pure rate control has the lowest reneging, but deviated rate control is close. Deviated rate control performs almost as well as pure rate control for the sudden increase in workload (shown in Figure 14) but better for the sudden increase in workload with hourly spikes (shown in Figure 12). The rate-based policy for multiple service classes also performs almost as well as pure rate control, but it has the advantage of being able to offer different service levels. The average wait times for the three rate-based policies are nearly the same and significantly better then the average for forced wait.

6 Capacity Planning

An important aspect in the design of a video delivery system is determining the number of channels needed to provide a certain level of service. Instead of defining a system and then simulating the performance, we use a simulator to determine the minimum number of channels needed to meet a long-term reneging objective of 5%, and a short-term reneging objective of 15%. Results presented in this section are generated by iteratively simulating larger and larger systems until the short- and/or long-term reneging objective are met.

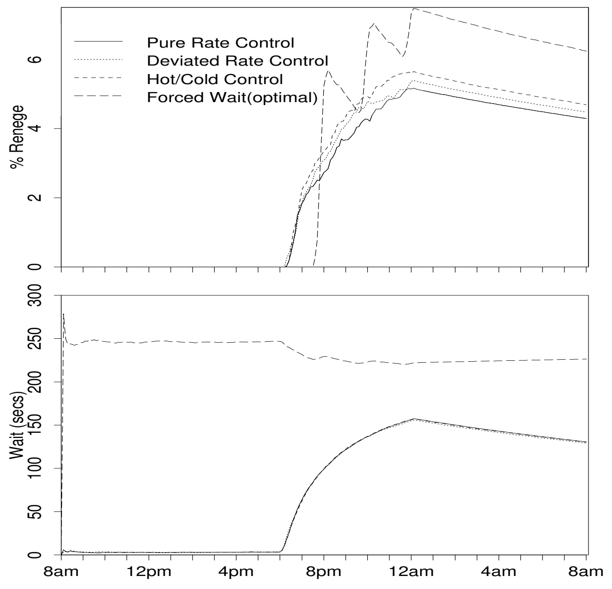

Figure 14: Comparison of channel allocation policies.

6.1 Long Term Reneging Objective

Figure 15 shows the number of channels needed to meet only the 5% long-term reneging objective. The workload used is the same as the one used for the other results presented in this paper. The x-axis represents the average number of requests arriving per minute for the non-prime time hours. During prime time, the average request arrival rate is 5 times greater.

The results in Figure 15 show that as the request arrival rate increases, the number of channels needed increases quickly for the system that cannot batch requests and more slowly for multicast-capable systems. In the system allowing batching but implementing only on-demand allocation, the number of channels needed increases rapidly but then levels off at approximately 3000. For the optimal forced wait and pure rate control policies the number of channels needed levels of about 1000 with pure rate control doing slightly better. Because the only requirement is long-term average reneging, forced wait performs nearly as well as pure rate control, but as Figure 6 shows, forced wait has occasional very high short-term reneging. Generally, in systems that allow batching, the number of channels needed levels off with increasing load since the the effective capacity increases because of increased batching.

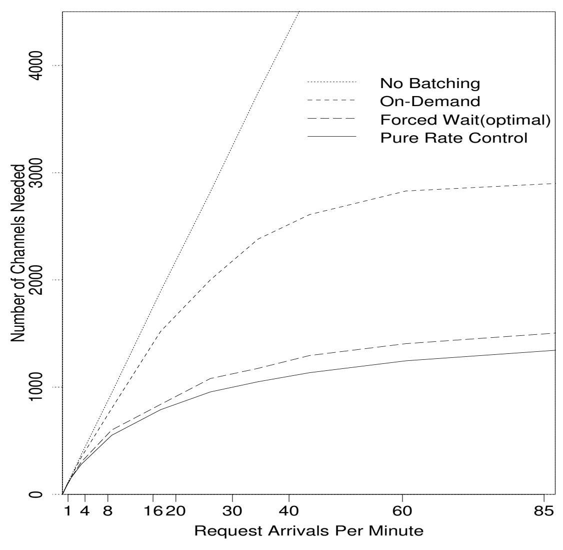

Figure 15: Required server capacity using a long term reneging objective only.

6.2 Short- and Long-Term Reneging Objective

Because predictable and consistent performance is as important as overall performance, both short- and long-term reneging objectives need to be considered. The results in Figure 16 show that for the no batching case, the slope of the line is slightly greater than in Figure 15 indicating that some additional channels are needed to eliminate cycles. For the on-demand policy, cycles are a significant problem and 25% more channels are needed to achieve both reneging objectives. The same holds true for the forced wait policy which requires more than twice as many channels. And finally, because pure rate control was designed to smooth peaks, additional channel requirements are negligible.

6.3 Channel Utilization

A final measure of performance is the ability of a video delivery system to achieve high channel utilization during both low and high load periods. High utilization is also desirable because it means per-movie costs are lower. The *effective capacity* of a system is defined as the number of requests being serviced divided by the system capacity. For video systems that provide batching the effective capacity may be greater than 100%. Figure 17 shows the effective capacity for three systems.

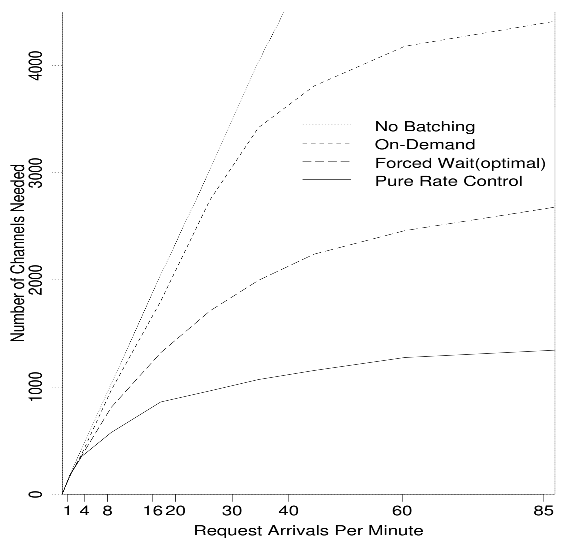

Figure 16: Required server capacity with short- and long-term reneging objectives.

The system that does not allow batching has a high average reneging and the lowest effective capacity. The effective capacity for systems that allow batching are much higher, but implementing a rate-based allocation policy increases the effective capacity even more. Pure rate control achieves additional gains over on-demand allocation because during high load periods reneging is much lower and more requests are serviced per channel.

7 Summary and Conclusions

In this paper, we identify through simulation the limitations of greedy channel allocation policies, especially in the context of video delivery systems with multicast/batching capability. These algorithms are poor solutions because: (1) they do not minimize customer waiting time, (2) they perform poorly when load conditions vary meaning they do not even achieve their goal of maximizing customers serviced, and (3) they have wild oscillations in short-term performance. To solve these limitations, we propose a set of rate-based policies that work by ensuring that channels are available for allocation on a consistent basis. Not only do these policies achieve a consistent and predictable level of service in terms of customer reneging, but they also improve effective capacity and resource utilization.

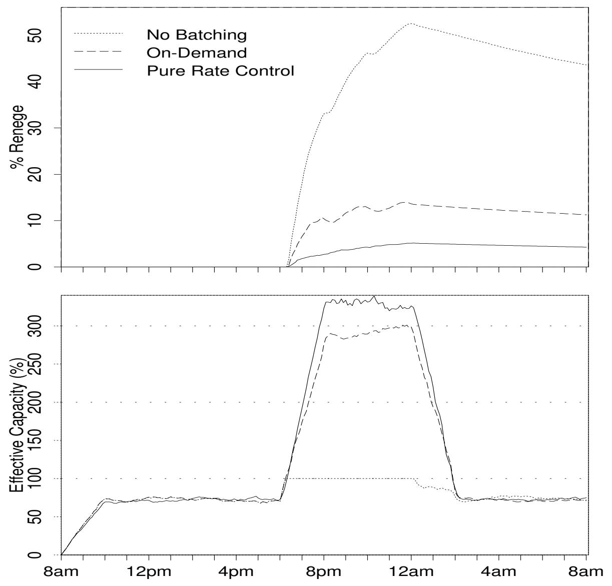

Figure 17: Channel utilization in terms of effective capacity.

In this paper, we offer three contributions. First, the development of workload models that are not steady state Poisson models and more accurately reflect customer behavior over a 24-hour period. Second, the use of a video delivery system model to simulate the performance and show the limitations of several greedy channel allocation algorithms. Third, the proposal of rate-based channel allocation algorithms that solve the problems of the greedy algorithms.

References

[1] D. Anderson, "Metascheduling for continuous media," *ACM Transactions on Computer Systems*, vol. 11, pp. 226–252, Aug 1993.

[2] V. Rangan, H. Vin, and S. Ramanathan, "Designing an on-demand multimedia service," *IEEE Communications Magazine*, vol. 30, pp. 56–64, Jul 1992.

[3] A. Campbell, G. Coulson, and H. D., "A quality of service architecture," *ACM Computer Communication Review*, Apr 1994.

[4] K. Ramakrishnan and et al., "Operating system support for a video-on-demand file service," *Multimedia Systems*, vol. 3, pp. 53–65, May 1995.

[5] J. Gemmel, H. Vin, D. Kandlur, V. Rangan, and L. Rowe, "Multimedia storage servers: A tutorial," *IEEE Computer*, pp. 40–49, May 1995.

[6] T. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, pp. 14–23, Fall 1994.

[7] A. Dan and D. Sitaram, "A generalized interval caching policy for mixed interactive and long video environments," Tech. Rep. RC 20206, IBM Research Division, T.J. Watson Research Center, 1995.

[8] A. Dan, D. Dias, R. Mukherjee, D. Sitaram, and R. Tewari, "Buffering and caching in large scale video servers," in *IEEE CompCon*, pp. 217–224, 1995.

[9] K. Almeroth and M. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas of Communication*, Aug 1996.

[10] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *ACM Multimedia '94*, (San Francisco, CA), Oct 1994.

[11] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley, "Channel allocation under batching and vcr control in movie-on-demand servers," Tech. Rep. RC 19588, IBM Research Division, T.J. Watson Research Center, 1994.

[12] L. Golubchik, J. Lui, and R. Muntz, "Reducing i/o demand in video-on-demand storage servers," in *ACM SIGMETRICS '95*, May 1995.

[13] H. Dykeman, M. Ammar, and J. Wong, "Scheduling algorithms for videotex systems under broadcast delivery," in *ICC '86*, pp. 1847–1851, 1986.

[14] J. Wong and M. Ammar, "Analysis of broadcast delivery in a videotex system," *IEEE Transactions on Computers*, vol. 34, pp. 863–866, Sep 1985.

[15] J. Dey-Sricar, J. Salehi, J. Kurose, and D. Towsley, "Providing VCR capabilities in large-scale video servers," in *ACM Multimedia '94*, (San Francisco, CA), pp. 25–32, Oct 1994.

[16] A. Dan, P. Shahabuddin, D. Sitaram, and W. Tetzlaff, "Anticipatory scheduling with batching for video servers," Tech. Rep. RC 19640, IBM Research Division, T.J. Watson Research Center, 1994.

[17] *Video Store Magazine*, Dec 1992.

[18] G. Zipf, *Human Behavior and the Principle of Least Effort.* Reading, MA: Addison-Wesley, 1949.

[19] Z. Zhao, S. Panwar, and D. Towsley, "Queueing performance with impatient customers," in *IEEE Infocom*, pp. 400–409, 1991.