RC 20249 (89566) 10/30/95
Computer Science/Mathematics 15 pages

Research Report

Long Term Channel Allocation Strategies for Video Applications

Kevin C. Almeroth

Networking and Telecommunications Group
Georgia Institute of Technology
Atlanta, GA 30332-0280

Asit Dan, Dinkar Sitaram and William H. Tetzlaff

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

Long Term Channel Allocation Strategies for Video Applications[1] [2]

Kevin C. Almeroth
Networking and Telecommunications Group
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Asit Dan
Dinkar Sitaram
William H. Tetzlaff
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

In a multimedia environment offering customers the ability to select and watch programs, service is expected to be nearly immediate and continuous. A video server can provide this type of service by reserving sufficient server and network resources (referred to as *logical channels*) for the duration of the program. In such systems, finite resources mean there is a hard limit on the number of programs that can be served concurrently. However, a single channel can be used to serve multiple customers waiting for the same program (referred to as *batching*). Batching is especially useful during high load periods (e.g., prime time) because it increases effective system capacity. However, without any control on the *channel allocation rate*, even slight variations in system load may introduce a significant problem shared by both batching and non-batching systems alike. Because channels are typically allocated as needed, all free channels may be allocated in a short period of time satisfying only a small percentage of waiting requests. Subsequent requests would then have to wait a very long period of time before channels again become available. This type of instability, or *cyclical* behavior, is undesirable, especially if one of the performance objectives is to provide a consistent level of service.

In this paper, we identify the conditions which lead to the development of *cycles* in the channel allocation rate. We then propose a rate-based allocation strategy which eliminates the cyclical behavior while improving effective capacity and reducing customer waiting time. Performance of video servers implementing several different channel allocation strategies are compared. The rate-based policies are shown to be robust for various workloads, and are easy to implement.

[1] Georgia Tech Technical Report GIT-CC-95-45
[2] IBM Research Report RC 20249

1 Introduction

In a video application environment, a system needs to reserve various resources needed for playout of a stream (referred to as a *logical channel*) in order to guarantee continuous delivery of video data[4, 17, 23, 5, 18, 22]. Hence, depending upon the server configuration, there is a hard limit on the number of streams that can be simultaneously delivered by a server[19, 21, 9, 10, 7, 8]. However, in the presence of a multicast facility [3], multiple customers requesting the same program within a short interval of time can be served by the same data stream. This policy, referred to as *batching*, takes advantage of the fact that there is a high frequency of access to popular programs and can result in substantial savings in server capacity [1, 2, 11, 12, 20, 15].

The arrival rate of program requests to such a system may vary with the time of day [13, 21]. For example, during prime time, the request rate may be much higher than the request rate during the early morning. Under an opportunistic scheduling policy that allocates channels on demand, the server resources will be exhausted at the beginning of a rising load period, since each customer will be allocated a channel of its own. Video playout time may be long (typically 2 hours) for such applications, resulting in unavailability of these channels for a long period of time. Therefore, very few channels (i.e. only the newly freed channels) will be available during this period of time. A waiting customer may renege if it has waited too long [6, 26, 11, 12]. Hence, this will result in high rejection rate of customer requests. In addition, these allocated channels may be freed all at once (within a short time period if the service times are nearly the same). This will lead to another period of high channel allocation followed by another period of low channel availability and high rejection. Such cyclic behavior is extremely undesirable, not only for the high rejection probability during peak period but also the non-uniform rejection probability throughout the day. The problem can only be alleviated by controlling the allocation rate, and making sure that the channels are conserved for peak periods[13].

We proposed a two level hierarchical scheduler in [13] where the first level controls the channel allocation rate in anticipation of the future load, and the second level selects waiting customers to be served with an available channel so as to reduce the overall customer reneging. In this paper, we propose various practical scheduling policies based on the concept of hierarchical scheduling, and explore in depth the robustness in performance of such policies.

Practical policies have to rely on incomplete and/or inaccurate knowledge of the future load as well as customer reneging behavior[11]. Generally, the maximum waiting time before an individual customer reneges is not known in advance. Even the statistical customer reneging behavior may be difficult to estimate without causing a large number of customers to renege (which is a poor design). Accurate information on future load is also not available in general. Hence, scheduling policies that depend heavily on a detailed knowledge of customer reneging behavior and/or future load are not practical. Single tier policies (even complex policies using the queue status) optimise allocation of channels only across currently waiting requests, and hence, can not avoid high short term reneging once channels are depleted.

The remainder of the paper is as follows. In section 2, we describe the architecture of a video application system with special attention to a two-tier scheduler. The system environment parameters, workload including user behavior, and policy objectives are described in detail in Section 3. In Section 4, we study the performance of two intuitive non-rate-based allocation policies (i.e., on-demand allocation and forced wait policies), and show that rate control is necessary. Subsequently, in Section 5, we propose a simple rate control policy and various extensions. To further demonstrate the performance advantages of rate-based policies, we contrast the required server capacities for a given reneging objective under both types of policies in Section 6. Section 7 contains our concluding observations.

2 Architecture

Video application systems can vary in size and complexity depending on several factors including the type of service to be offered, and the number of simultaneous streams the system is capable of providing. 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 the unnecessary details of a specific VOD system, a generic architecture with three main components is used. The operation of such a system is described in this section. Also, the *Scheduler* in the video server, where the channel allocation policies will be implemented, is described in additional detail.

2.1 Video System Architecture

Figure 1 shows a generic video application architecture with three main components: video server, network, and set of clients/customers. The typical operation of such a system is driven by customers making program requests to the video server. These requests flow over a two-way channel established for the purpose of exchanging control messages. A request to the video server is received by the control server which is responsible for attempting to satisfy the request given finite system resources. In the control server, the scheduler is responsible for allocating sufficient resources, both in the network and video server, to provide the requested service. The resources needed to deliver a single stream are referred to as a *logical channel*. The scheduler allocates these logical channels using a wide variety of policies ranging from simple on-demand strategies to more sophisticated rate-control strategies.

The more sophisticated scheduling policies proposed in this paper are designed to take advantage of an important characteristic of the system architecture. If several customer requests for the same movie occur in some interval of time they can be *batched* together and one logical channel can be used to service all of them. A scheduling policy which takes advantage of batching by delaying the start of some requests can simultaneously service more customers than logical channels. Increasing a system's *effective capacity* to greater than 100% is one goal of using a sophisticated scheduling policy.

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 (or additional bandwidth) has to be allocated upon resume (or other op-

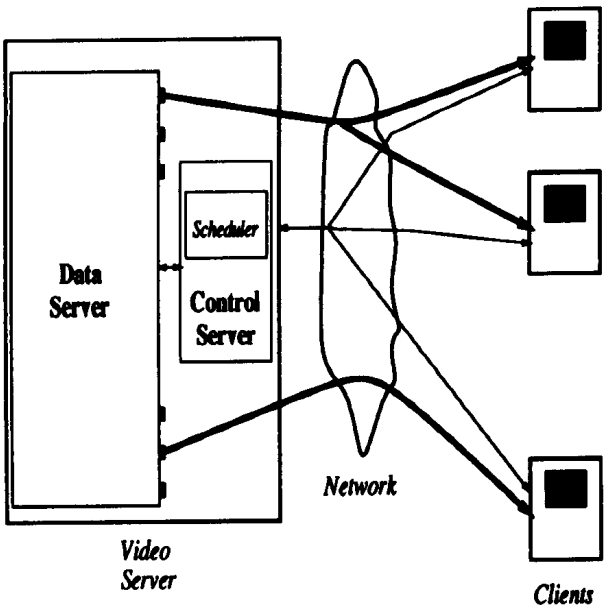

Figure 1: System architecture

eration) since the paused customer is no longer synchronised with the original batched stream (if any). In [1, 2, 11, 12, 14], it is proposed that some channels/bandwidth should be set aside to provide continuous service for such unpredictable VCR operations. These channels are referred to as *contingency channels*. In this paper, we consider only workloads without such VCR control operations, since the contingency channel pool under batching is an orthogonal policy issue. We will describe in the next subsection how the concept of contingency channels can be integrated with a two-tier scheduler.

2.2 Channel Scheduling Architecture

Figure 2 shows the components of the scheduler, and their interactions with the video player. The scheduler includes two components, a High Level Scheduler (HLS) and a Low Level Scheduler (LLS). Signaling between the HLS and LLS and between the scheduler and other modules is accomplished by setting and resetting flags or status bits indicative of various conditions and requests.

The HLS maintains a flag, CHAN_REQ_FLAG, and a counter for the number of free channels, FREE_CHAN_CNT. Client requests for the start of a new program are queued in the REQ_QUEUE of the LLS. The LLS sends "Channel Request" signal to the HLS to schedule waiting customer requests for start of a new program. The HLS sends signal "Channel Allocation" back to LLS once it allocates a channel. The LLS then selects a program to play and batches all waiting customer requests for that video. It then sends the signal "Play Program" to the video player.

The selection of which program to play depends on various attributes of the customer requests waiting in the queue, policy objectives, as well as the supported service class. Client request attributes include, for example, the amount of time each customer has waited, their reneging time threshold (how long each customer is expected to be willing to wait), if known, and the expected service time requirement for the program (i.e. how long the system resources are expected to be needed for playing of the program). The policy objective can be, for example, to minimise overall reneging, fairness, or prioritising one class of requests over others. For example, one requester might receive deluxe or preferred class service for a higher fee and thus be served on a priority basis.

In this paper, we will assume the LLS policy is FCFS (first come first serve), since it is shown to be effective in [11] and is also easy to implement. In accordance with FCFS, the request in the front of the queue is served first and all other customer waiting for the same program are batched together when the request is serviced. Thus, FCFS schedules programs based on the attributes of (1) position in the queue and (2) which program has been requested. Special treatment can be given to "hot" (popular) programs. In accordance with this policy (sometimes referred to as FCFS-n), hot programs are served from a special queue provided for that purpose. FCFS with a priority queue can be used for multiple service classes.

To support good service for VCR control operations, contingency channels/bandwidth as described in [1, 2, 11, 12, 14] are set aside by the HLS. The HLS controls channel allocation not only in anticipation of future load, but also for setting aside the required number of contingency channels.

3 Video System Environment

The system environment is characterized by parameters for (1) the system's size and operation, and (2) the system's customer behavior. The performance of a system so characterized is then measured in terms of several performance objectives. This section describes parameters in each of these areas.

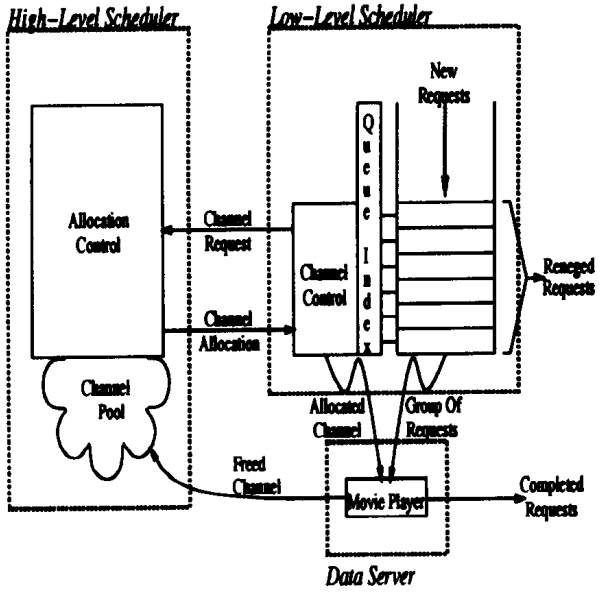

Figure 2: Channel scheduling architecture

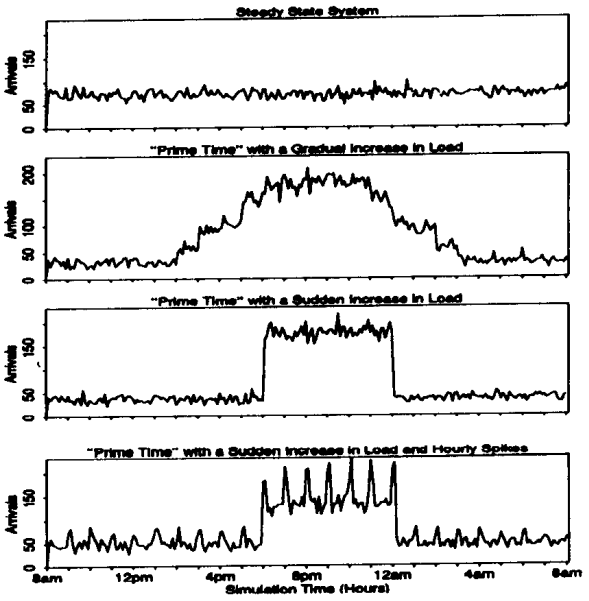

Figure 3: Various request arrival patterns

3.1 System Characteristics

The system characteristics define the size and operation of the video server. The parameters are:

1. **Number of Logical Channels**: Number of simultaneous movie streams that can be supported. The results presented in this paper are for a system with 1000 logical channels.
2. **Number of Offered Movies**: A customer makes a selection from a list of offered movies. Results are presented for systems with 100 movies.
3. **With or Without Batching**: Batching, if implemented by the system, allows multiple requests to be serviced with one channel. Simulations are run for both types of systems.
4. **Allocation Control Policy**: Various policies can be implemented in the video server. Details and and performance analysis for several policies are given in later sections.

3.2 User Behavior

User behavior includes all parameters which determine the type and frequency of customer requests. The parameters are:

1. **Request Arrival Pattern**: Simulations are run for a 24 hour period from 8am of one day to 8pm 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. Figure 3 shows four different types of workloads. The horisontal axis is time and the vertical axis is 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. For a 24 hour period this is unrealistic because demand fluctuations have been shown to exist[24]. A much more realistic model assumes a prime time period between 6pm and 12am where the prime time workload is five times greater than the rest of the day. The second and third parts of 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 exponential functions with different inter-arrival time averages. As we will show, the performance of these two workloads is very similar. The fourth workload is a special case with load surges at hour intervals. This workload is reasonable in a system with pre-scheduled broadcast programming starting and ending on the hour.

2. **Movie Selection**: For each request, a movie is selected using a Zipf distribution[27] with a parameter of 0.271. This distribution fits empirical evidence presented in [24]. The Zipf distribution roughly translates to an above average number of requests for "hot" movies and less frequent requests for "cold" movies.

3. **Reneging Behavior**: The simulated reneging behavior is that any customer who makes a request will wait a minimum of 5 minutes plus a variable wait time exponentially distributed with an average of 2.5 minutes.

4. **Service Time**: The service time is defined as 2 hours plus or minus 10 minutes. This reflects the variability in the length of different programs, as well as variable viewing time due to VCR operations for unbatched programs.

3.3 Performance Objectives

The performance objectives are defined in terms of three performance measures which are:

1. **Average Waiting Time**: The time from when a customer makes a request to the time the movie starts should be minimised. Furthermore, waiting time should be consistent and increase gracefully during high loads.

2. **Long Term Average Reneging**: Over the entire simulation period the number of customers who leave because they have have waited too long for a movie to start should be minimized.

3. **Short Term Peak Reneging**: Reneging measured over short intervals (6 minutes) should not vary significantly. Reneging may increase slightly during sustained high load periods, but it should not oscillate wildly between intervals.

Relatively higher values of short term reneging may be acceptable. In this paper, we assume that acceptable long term reneging is 5%, and acceptable peak value for short term reneging is 15%. The conclusions and relative orderings of the policies are not dependent on these parameters.

4 No Allocation Control

In this section, we describe the operation and performance of video servers which do not control the rate of channel allocation. These systems perform poorly under *high load* conditions, i.e. when the request arrival rate is higher than the service/consumption rate. Poor performance is characterised by (1) high long-term average reneging, and (2) the development of cycles alternating between very high and very low short term reneging.

The two policies discussed are "on-demand allocation" and "forced wait". While neither policy controls the rate of allocation, both do implement opportunistic batching. All requests for the same movie that are in the queue when a channel is allocated are batched together. The on-demand allocation policy allocates channels as soon as a request is made while the forced wait policy delays the allocation of a channel to increase the likelihood of batching. 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 for allocating channels is:

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

2. When a channel is freed, the scheduler immediately checks to see if there are any requests waiting. If such a request exists, it is immediately allocated the free channel.

Requests are serviced first come, first served. During low loads there will usually be at least one free channel each time a request arrives. This means no requests will be queued, and no batching will occur. During high load periods requests are frequently queued and batching is more common. However, during sustained high load periods, reneging occurs frequently.

Figure 4 is the first of a series of time dependent graphs which are used to show the state and performance of a video server implementing a particular scheduling policy. The horizontal axis is time and starts with to 8am of one day and extends to 8am of the next. Each point in the graph, called a *reporting interval*, represents an average taken over the previous 6 minute. The dotted lines represent cumulative averages computed at the end of each interval. The parts of these graphs are:

1. **Arrivals**: The number of requests that arrived during a reporting interval.
2. **% Renege**: The percentage of requests that were cancelled or reneged because the customer waited too long.
3. **Channels Alloc**: The number of channels allocated during a reporting interval.
4. **Wait(secs)**: The average wait time of all successfully started requests.
5. **In Queue**: The average number of requests waiting for service.

Figure 4 shows the performance of a VOD system that implements on-demand allocation of channels. This system suffers from poor long term performance and high peak reneging. During the prime time hours, there are three cycles in reneging. High reneging occurs because channels are allocated more quickly than they are being freed. When the load jumps at 6pm, there are only enough channels to satisfy requests on an on-demand basis for a half hour. After a half hour, all free channels are completely exhausted. Furthermore, since most of the channel pool was just allocated, these channels will not be available for approximately another two hours. Until these channels are freed a majority of new requests will be blocked. When this group of channels eventually becomes free they will again be completely allocated in a short period of time. This behavior repeats itself as long as the load remains high.

Figure 4 also shows how the state of the system varies with load conditions. First, when no channels are available the number of requests in the queue increases quickly, and the average wait time of requests that do get satisfied goes from zero to almost five minutes. During sustained high load periods, customers are almost always blocked and the average wait time oscillates wildly. This type of system performance is unacceptable.

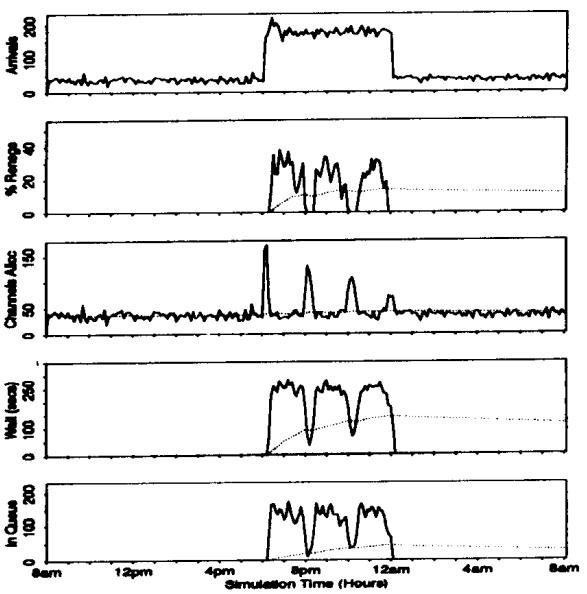

Figure 4: On-demand allocation with a sudden increase in load.

To gain more understanding of this cyclical behavior, Figure 5 shows an enlargement of the time period from 6pm to 11pm. The additional detail shows more clearly the cyclical performance and, more importantly, the duration of high reneging, and long wait periods. For every two hour cycle, the system has acceptable behavior for only 25% of the time. For the other 75% of the time, requests average reneging is more than 50%.

Figure 6 shows the performance of a video server implementing the on-demand allocation policy under a different workloads; one that increases more slowly to a high load. The results are similar to those in Figure 4. Results shows that cycles develop almost immediately after the load starts to increase. In general, cycles occur when a large percentage of the channel pool is allocated in a short period of time.

4.2 Forced Wait

Forced wait is a scheduling policy which requires at least one of the requests to have waited some minimum amount of time. Ideally, this *minimum wait time* is a tradeoff between maximizing customer waiting times and minimizing the probability of customer reneging. The goal is to force requests to wait as long as possible thereby increasing the batching factor and effective server capacity. Forced wait may

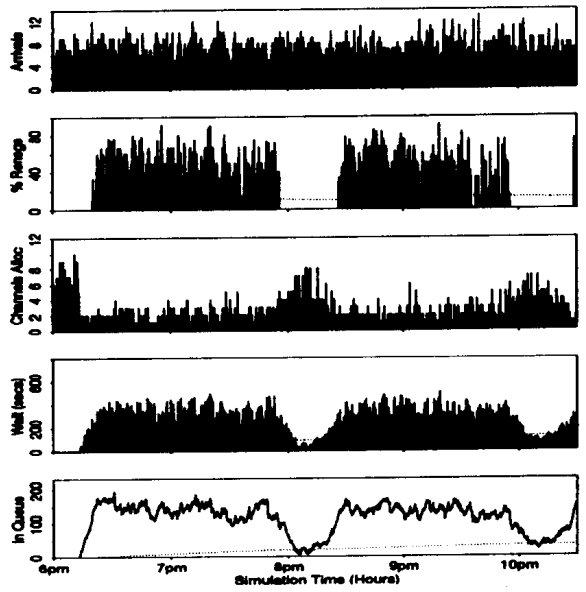

Figure 5: Enlargement of cycles in a system using on-demand allocation.

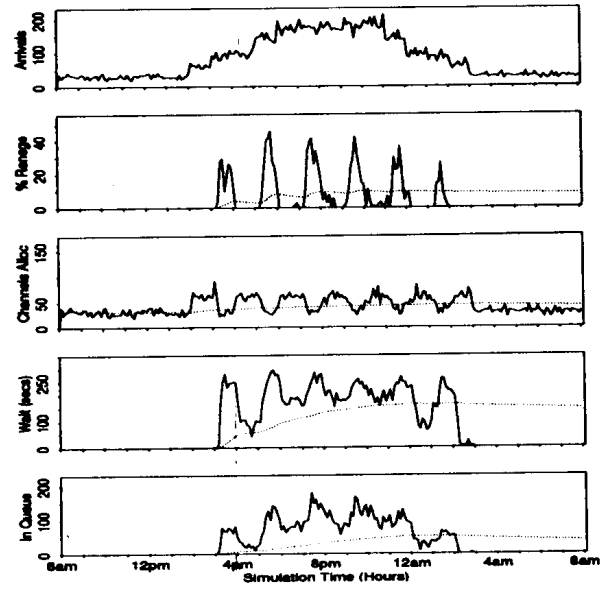

Figure 6: On-demand allocation with gradual increase in load.

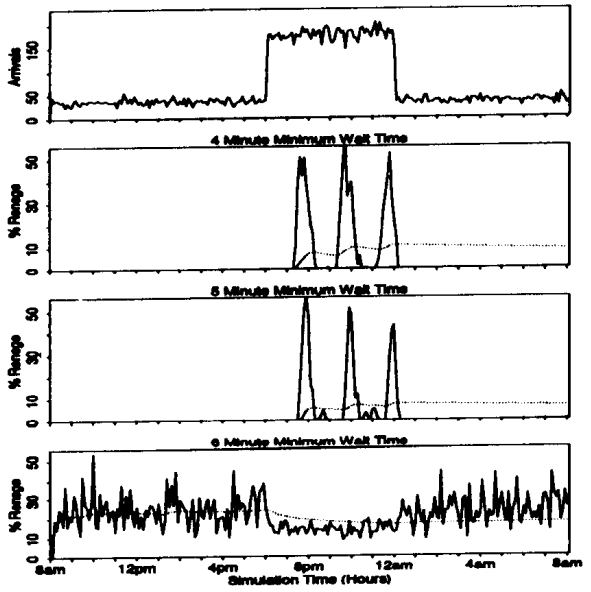

Figure 7: Forced wait with various minimum wait times.

delay the allocation of channels, but it does not attempt to control the allocation rate. The low-level scheduler simply holds the first request in the queue for the minimum wait time before requesting a channel for allocation. Once this happens, a channel is immediately requested from the high-level scheduler. There are several drawbacks to using forced wait.

One major drawback of the forced wait policy is that the minimum wait time parameter must be set by a system operator. As reneging behavior changes over time, so will the performance of the system. Unless this parameter is chosen correctly and dynamically adjusted reneging will be unacceptably high. Figure 7 shows the reneging levels for the same VOD 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. Through simulation, the optimal minimum wait time value was determined to be just slightly more than 5 minutes. If this value is too high, reneging increases by almost 300%. Choosing a value that is too low increases reneging by nearly 50%. A value that is too high forces customers to wait longer than they are willing even though the system could have allocated them a channel. A value that is too low does not achieve optimal batching.

A second major drawback of forced wait is that it does not fix the problem of cyclical behavior even with an optimal minimum wait time. Figure 8 shows

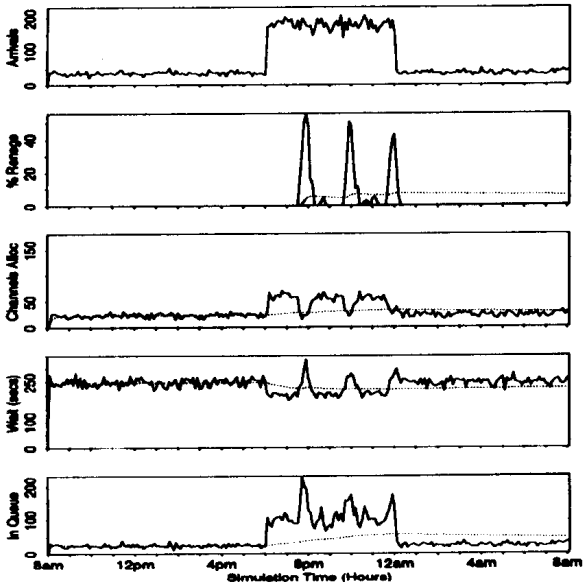

Figure 8: Forced wait with optimal minimum wait time.

the performance of a forced wait system with the optimal minimum wait time(5 minutes). The results exhibit similar characteristics 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, reneging is unacceptable for almost half of each cycle.

A third major drawback of forced wait is the high waiting time experienced by each customer. Even during low loads, many customers are forced to wait the full minimum waiting time. This policy is certainly not appropriate for systems which 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 batching is low so there is no advantage to making all customers wait.

5 Rate-Based Allocation

The two policies analyzed in the last section suffer from poor and inconsistent performance. A solution is to control the rate at which channels are allocated using a rate-based policy in the high-level scheduler. In this section we propose a basic rate-based policy and show its performance. We also describe two modifications which can be made to the basic policy.

5.1 Pure Rate Control

The pure rate control policy (1) puts an 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 *measurement interval.* The purpose of the interval is to provide a time period over which to accurately calculate and enforce the rate control policy. 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 earliest time when a channel can be allocated again. T_{next} is given by $T_{next} = T_{last} + \Delta$ where Δ is the time interval between successive allocations. Let T_{left} be the remaining time until the end of the interval, C_{max} be the maximum number of channels that can be allocated during this interval, and C_{alloc} be the number of channels allocated so far. 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 system, C_{total}. Let T_{int} be the length of the measurement interval. Then C_{max} is computed as $C_{max} = (T_{int} * C_{total})/T_{view}$. The video server simulated in this paper is characterised 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 9. This limit controls the channel allocation rate within a measurement 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 inter-arrival time of requests will exceed the delay imposed between channel allocations. During low loads requests will not experience any delay. Figure 9 shows two such periods.
- When the request arrival pattern is high enough, requests will arrive more quickly than channels can be allocated. Requests will be

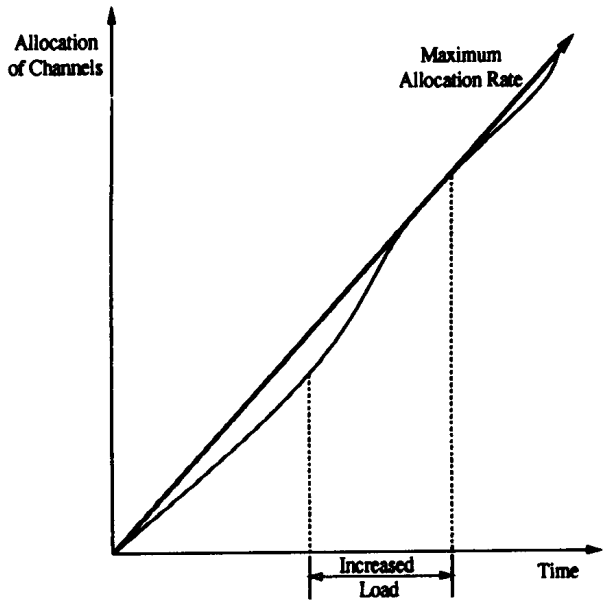

Figure 9: Allocation rate limits for pure rate control.

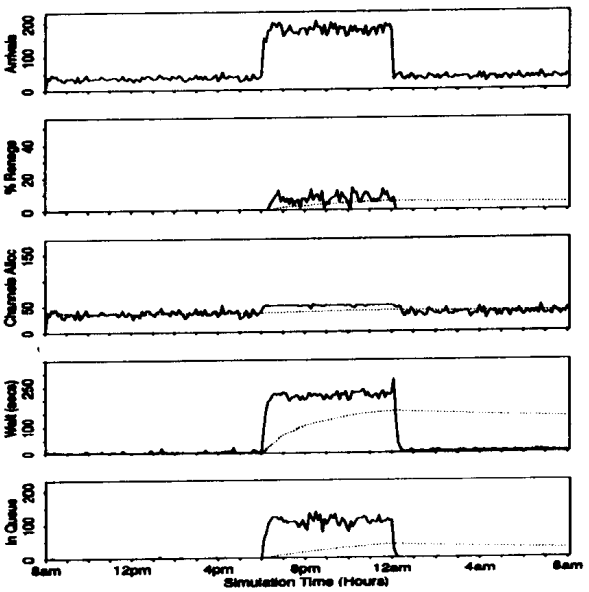

Figure 10: Pure rate control with a sudden load increase.

queued and must wait for the high-level scheduler to allocate a channel. If no channel is available, requests will remain queued and the next channel freed will be allocated immediately.

- When the workload transitions from low to high load, the allocation rate can be increased up to the maximum allocation rate. Figure 9 shows such an example. When the increased load period begins, the number of allocated channels is somewhat below the maximum. The high-level scheduler allocates channels at a slightly increased rate up to the maximum were it is forced to level out. This mechanism allows pure rate control some flexibility in adapting to changes in workload. In the next section we will look at relaxing this control to achieve additional flexibility.

Figure 10 shows the performance of a video server that implements pure rate control. The results show that pure rate control solves all of the problems of on-demand allocation and forced wait. 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 a higher 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. Both aspects are critical to satisfactory performance.

Figure 11 shows the performance of the same system, but for a workload with a gradual increase to a high load. The results are almost the same as the results with the first workload. Pure rate control has been shown to be an improvement over the other policies.

5.1.1 Advantages of Rate Control

Control of channel allocation using a rate-based combines the best characteristics of the on-demand and forced wait policies. The results in Figure 12 compare the 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 with an optimal minimum wait time. The second part of the graph shows that the average wait time for pure rate control is very close to being the lowest. Only on-demand allocation is better, but of course it suffers from high average reneging and performance cycles. Finally, pure rate control has the advantage that it does not require any parameters to be set by a system operator.

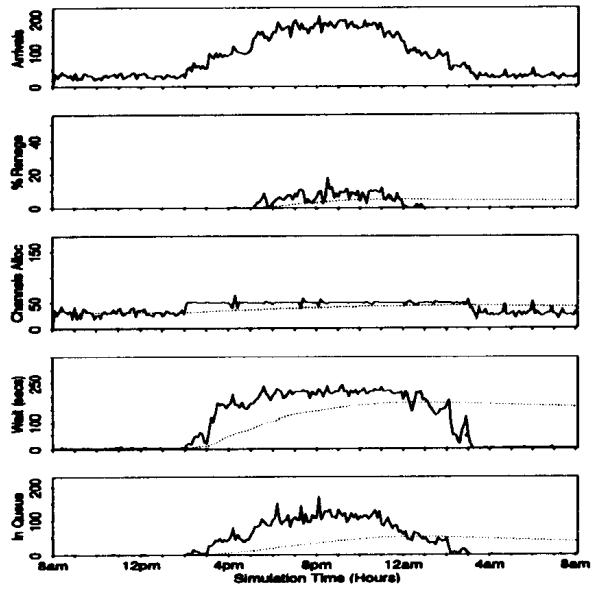

Figure 11: Pure rate control with a gradual load increase.

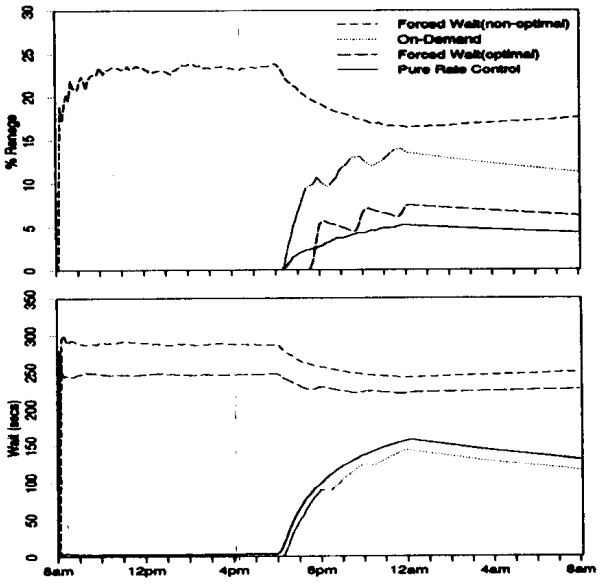

Figure 12: Comparison of channel allocation policies.

5.2 Extension 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 for some workloads while maintaining the performance level of pure rate control for the rest. The second extension looks at ways in which different levels of service can be provided for different categories of movies.

5.2.1 Deviated Rate Control

Deviated rate control is similar to pure rate control. The difference is that under certain conditions deviated rate control allows the consumption rate to temporarily rise above the maximum allocation rate. However, the maximum number of channels that can be allocated during a measurement interval is still the same. Deviated rate control simply provides more flexibility in allocating this pre-determined number of channels. This additional flexibility is an important feature which can further reduce reneging for some workloads. Instead of forcing the channel allocation rate to stay at or below the maximum allocation rate, deviated rate control allows the rate to exceed this level by an operator-defined value.

Deviated rate control requires two parameters to be defined, (1) the *maximum allowed deviation*, and (2) a *minimum wait time* 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 close to reneging. Figure 13 shows such an example. The maximum allowed deviation parameter can be used to implement a rate-based policy with any level of control. A value of one is the same as pure rate control. A value of zero is the same as forced wait except that the total number of channels allocated per measurement interval is still limited.

Figure 14 compares the performance of the rate con-

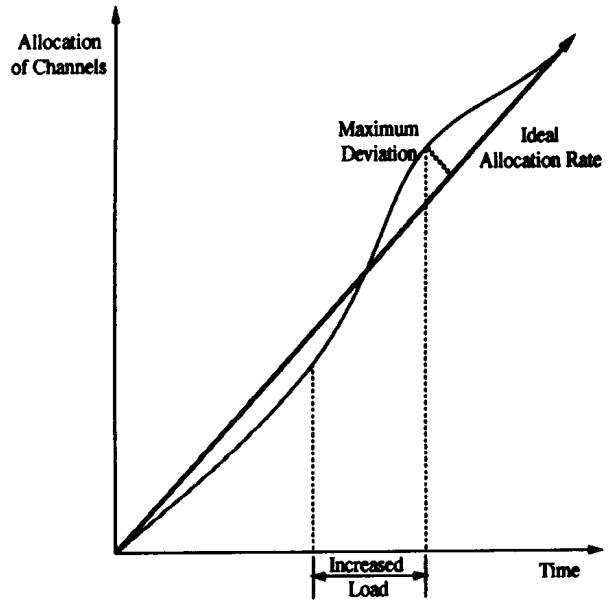

Figure 13: Allocation rate limits for deviated rate control.

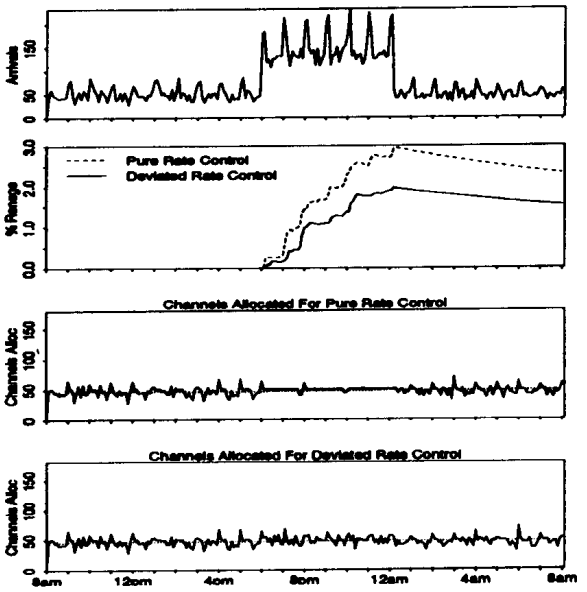

Figure 14: Deviated rate control

trol schemes for a workload with demand spikes on each hour. This alternate type of workload demonstrates a scenario in which deviated rate control is expected to and does perform better than pure rate control. As Figure 14 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 and the wait time threshold is 5 minutes. During the high load period the number of channels allocated by the pure rate control policy is relatively smooth, but for deviated rate control the number is less consistent. Allowing the allocation rate to deviate gives the high-level scheduler greater ability to adjust to short term surges in demand.

5.2.2 Multiple Service Classes

An allocation rate control policy using multiple service classes provides a mechanism to offer a variety of service classes. The ability to offer different levels of service for different prices is an important consideration in the design of a video server. An example of two service classes are:

- **Hot or Popular Movies:** Movies designated as hot are given a higher priority and the waiting time is limited to five minutes.
- **Cold or Unpopular Movies:** Movies that are not requested frequently are given a lower priority and the movie is not guaranteed to start within 5 minutes.

A variety of methods can be used to determine which movies are hot and which movies are not. One example is to statically chose some number to be hot. Another example, and the method used in this paper, is to use the last measurement interval's request count and a threshold. Any movie with enough requests is considered hot. The class of a movie is used by the high-level scheduler to determine the type of allocation policy to use for the request. The policies for the two service classes in this example are:

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 that movie satisfied. Hot movies have scheduling priority over cold movies.
2. **Cold Movies:** Scheduling for cold movies is done using pure rate control. However, any channels allocated for hot movies are still counted against the total number of channels which can be allocated in a measurement interval. Starvation is a possibility but unlikely since only a few movies are typically hot.

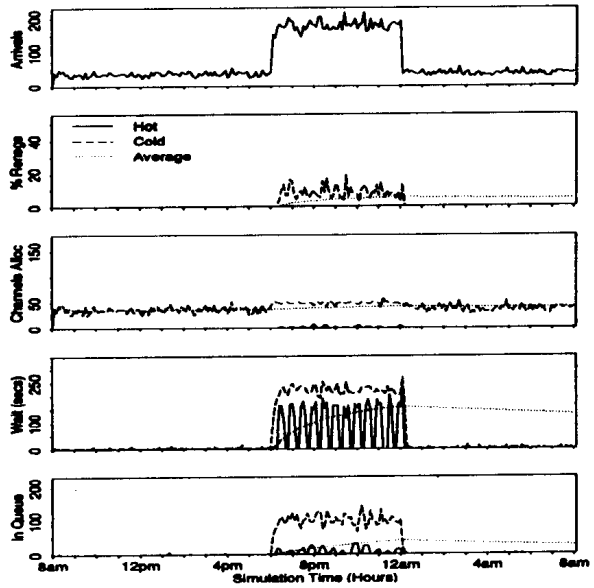

Figure 15: Allocation based on multiple service classes.

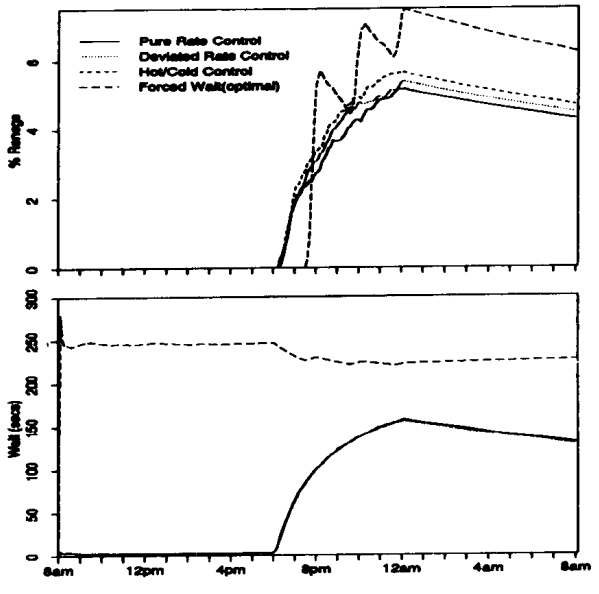

Figure 16: Comparison of channel allocation policies.

Figure 15 shows the performance of an allocation rate control policy using multiple service classes. Only during the high load period were there enough requests to make any of the movies hot. During the entire simulation no request for hot movies reneged. Because only a few movies were hot, only a few channels needed to be allocated for these movies. Furthermore, only a small percentage of the queue was actually hot movie requests. And finally, the average wait time was slightly better for hot movies since a limit was put on the waiting time and there was no rate control for hot movies.

5.3 Comparison of Allocation Policies

In this section we compare the long term averages of three rate-based allocation control policies and the forced wait policy with an optimal minimum wait time. Figure 16 shows that all three rate-based policies have lower reneging and much better average wait times. 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 16) but better for the sudden increase in workload with hourly spikes (shown in Figure 14). 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 server is determining how many logical channels are needed to achieve a given reneging objective for a given workload. This problem is much more complex for systems expected to handle variations in workload. A tradeoff must be made between building a system with enough resources to handle high loads, but not too many resources so that underutilisation occurs during low loads. The two performance objectives which will be used are (1) long term average reneging and (2) short term peak reneging.

6.1 Long Term Reneging Objective

The average long term reneging objective for the results shown in Figure 19 was 5%. The results in Figure 19 show that as the request arrival rate increases, the number of channels needed to meet the objective in a video server that does not allow request batching grows very rapidly. In the system which allows batching but implements only an on-demand

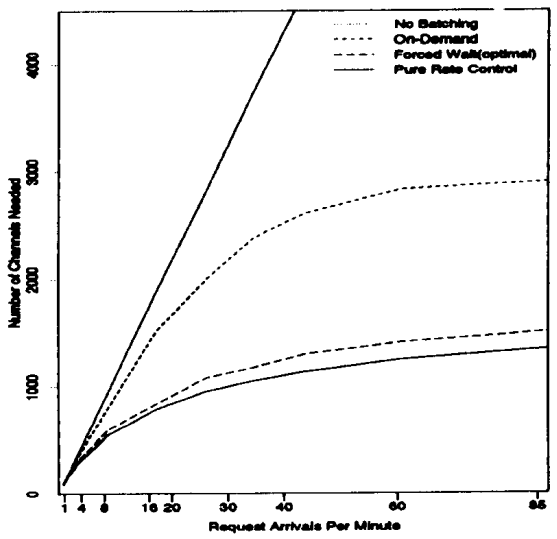

Figure 17: Required server capacity using a long term reneging objective only.

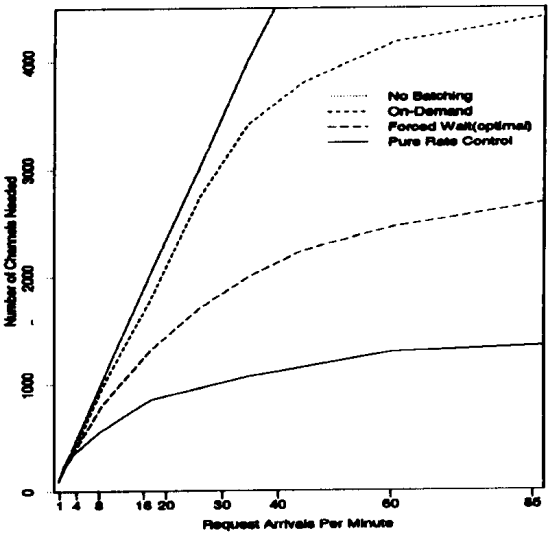

Figure 18: Required server capacity using long term and peak reneging objectives.

allocation policy the number of channels needed increases rapidly but then levels off at approximately 3000 channels. 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. Generally in systems with batching, the number of channels needed levels off with increasing load since at higher loads all requests are batched.

6.2 Peak Reneging Objective

A short term peak reneging objective is used to account for cycles that can develop during high load periods. The objective used in the results in Figure 18 is 15%. This means that the average reneging during each reporting interval must be below 15%. The long term average reneging must be below 5%. The results in Figure 18 show that for the no batching case, the slope of the line is slightly greater than in Figure 17 indicating that there were some cycles that additional channels eliminated. For the case of on-demand allocation in a server that allowed batching, cycles were a significant problem and 25% more channels were needed to achieve both reneging objectives. The same holds true for the system implementing forced wait with an optimal minimum wait time. More than twice as many channels are needed to achieve the short term peak reneging objective. And finally, because pure rate control was designed to smooth peaks, the difference between these curves in Figures 17 and 18 is negligible. Pure rate control achieves both objectives for the same video server while the only solution for other systems is to increase video server size and cost.

6.3 Channel Utilization

A final measure of performance is the ability of a video server to achieve high channel utilisation during both high and low load periods. High utilisation is also desirable because it means per-movie costs can be kept low. The *effective capacity* of a system is defined as the number of requests serviced at any instant in time. For video servers that provide a batching mechanism the effective capacity may be greater than 100%. Figure 19 shows the effective capacity at each reporting interval for three systems. The system that does not allow batching has a high average reneging and an effective capacity that can only equal 100%. The effective capacity for systems that do allow batching are much higher, but implementing a rate-based allocation policy increases the effective capacity still further. 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.

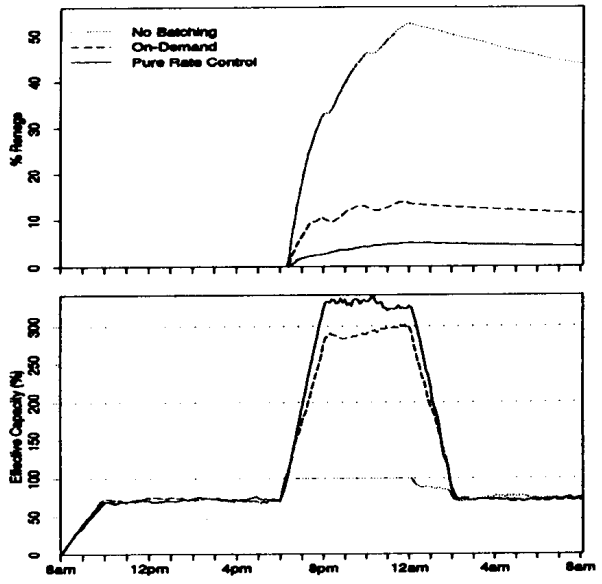

Figure 19: Channel utilization in terms of effective capacity.

7 Summary and Conclusions

In a multimedia environment, to guarantee continuous delivery for the duration of a program, the system reserves sufficient resources (referred to as logical channels) before starting playout of the requested program. Hence there is a hard limit on the number of programs that can be served concurrently by a given server. However, a single stream can be used to serve multiple customer waiting for the same program (referred to as batching). The system load can vary greatly from hour to hour (e.g., prime time vs late night). Hence, if channels are allocated as soon as they are available without any regard for future load, a large percentage of channels may be allocated in a short time period. Since service times may be very long (e.g., 2 hours for a movie), very few channels (remaining small percentage of the total channels together with the newly freed channels) will be available for a long period of time. This will result in cyclic shortage of channels followed by inefficient usage of channels (individual customer holding their own channel). By purposely delaying the allocation of a channel to a group of waiting requests, batching can be increased and the channel consumption rate can be reduced. Channels are thus preserved for load surges in the near future.

In this paper, we explore two-level admission control policies where a high level component controls the channel consumption rate, and a low level component schedules the program for playback based on waiting requests, once a channel is made available by the high level scheduler. The policies that do not use an explicit channel allocation rate perform poorly due to periodic shortage of channels coupled with ineffective batching as explained above. Under an on-demand policy no attempt is made to conserve channel and to achieve effective batching. The effectiveness of batching can be improved by forcing a customer at the front of the queue to wait a minimum wait time before it is served. A customer may renege if it is made to wait too long. Hence, selection of optimal minimum wait time is rather difficult. The amount of time an individual customer is willing to wait before reneging is not known in advance. Even the statistical reneging behavior may be difficult to estimate without causing a large number of customers to renege (which is a poor design). Even with such knowledge of customer reneging behavior, the optimal minimum wait time is hard to estimate without accurate knowledge of future load. Too small a minimum wait time can cause cyclic behavior, i.e., temporary over-allocation followed by unavailability of channels. Too large a minimum wait time on the other hand will conserve too many channels and hence, keep them idle, while causing some customers to renege. The forced wait policy also makes customers wait unnecessarily at the time of low load and for "cold" programs.

We then demonstrated that the proposed simple allocation rate control policy that allocates channels at the same rate at all times irrespective of the load can avoid the periodic shortage of channels. The main strength of this policy is that it requires vary little information about the customer behavior or load. It is therefore easy to implement, and is robust. For a given bound on the customer reneging probability, under this policy the system needs fewer channels not only to bound overall reneging probability, but also to provide consistent service level at all times. The allocation rate control can be made flexible under a deviated rate control policy, where the temporary allocation is allowed to deviate from the ideal rate by a given threshold. This could further minimise reneging for certain workloads (e.g., small spikes in the arrival rate). Rate control can also be integrated with multiple service classes for different types of programs and/or customers. Finally, detailed knowledge of future loads, if known, can be exploited to better control the allocation rate as proposed in [13].

References

[1] Almeroth, K., and M. Ammar, "Providing a scalable, interactive video-on-demand service using multicast communication", *ICCCN*, September 1994.

[2] Almeroth, K., and M. Ammar, "On the performance of a multicast delivery video-on-demand service with discontinuous VCR actions", *ICC*, June 1995.

[3] Almeroth, K. and Ammar, M., "On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service," *Journal on Selected Areas of Communication*, April 1996.

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

[5] Campbell, A., G. Coulson, and D. Hutchison, "A Quality Of Service Architecture", *ACM Computer Communication Review*, April 1994.

[6] 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.

[7] Dan, A., M. Kienzle and D. Sitaram, "Dynamic Policy of Segment Replication for Load-Balancing in Video-on-Demand Servers", *ACM Multimedia Systems*, Vol. 3, No. 3, July 1995, pp. 93–103. Also *IBM Research Report, RC 19589*, Yorktown Heights, NY, 1994.

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

[9] Dan, A., and D. Sitaram, "A Generalised Interval Caching Policy for Mixed Interactive and Long Video Environments" *IBM Research Report RC 20206*, 1995.

[10] Dan, A., and D. Sitaram, "An Online Video Placement Policy based on Bandwidth to Space Ratio (BSR)", *Proc. SIGMOD'95*, San Jose, May 1995.

[11] Dan, A., D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching", *Second Annual ACM Multimedia Conference and Exposition*, San Francisco, CA, October, 1994. Also to appear in *ACM Multimedia Systems*.

[12] Dan, A., P. Shahabuddin, D. Sitaram, and D. Towsley, "Channel Allocation under Batching and VCR Control in Video-On-Demand Servers", To appear in the *Journal of Parallel and Distributed Computing*. Also *IBM Research Report RC 19588*, Yorktown Heights, NY, 1994.

[13] Dan, A., P. Shahabuddin, D. Sitaram, and W. Tetzlaff, "Anticipatory Scheduling with Batching for Video Servers", *IBM Research Report RC 19640*, Yorktown Heights, NY, 1994.

[14] Dey, J., J. Salehi, J. Kurose, and D. Towsley. Providing vcr capabilities in large-scale video servers. In *Proceedings of ACM Multimedia Conference*, pages 25–32, 1994.

[15] 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.

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

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

[18] Ghandeharizadeh, S., and L. Ramos, "Continuous Retrieval of Multimedia Data Using Parallelism", *IEEE TKDE*, Vol. 5, No. 4, August 1993.

[19] Gemmel, J., H. Vin, D. Kandlur, V. Rangan, L. Rowe, "Multimedia Storage Servers: A Tutorial", *IEEE Computer*, May 1995, pp. 40-49.

[20] Golubchik, L., J. C.-S. Lui, and R. R. Munts Reducing I/O Demand in Video-On-Demand storage Servers In *Proceedings of ACM SIGMETRICS/ Performance '95*, Ottawa, Canada, May 1995.

[21] Venkatesh, D., and T. D. C. Little, "Prospects for Interactive Video-on-Demand", *IEEE Multimedia*, Fall 1994, pp. 14-23.

[22] Ramakrishnan, K. K., L. Vaitsblit, C. Gray, U. Vahalia, D. Ting, P. Tzelnic, S. Glaser, W. Duso "Operating System Support for a Video-On-Demand File Service." *Multimedia Systems*, Vol. 3 No. 2, May 1995, pp.53–65.

[23] Rangan, P. V., H. M. Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service," *IEEE Communication Magazine*, Vol. 30, July 1992, pp. 56–64.

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

[25] 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.

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

[27] Zipf, G., "Human Behavior and the Principle of Least Effort", Addison-Wesley, Reading, Mass, 1949.