On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service *

Kevin C. Almeroth
Mostafa H. Ammar

Networking and Telecommunications Group
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{kevin, ammar}@cc.gatech.edu
(404)894-3292

January 10, 1996

Abstract

In typical proposals for Video-On-Demand (VOD) systems, customers are serviced individually by allocating and dedicating a transmission channel and a set of server resources to each customer. This approach leads to an expensive-to-operate, non-scalable system. We consider a VOD system that uses multicast delivery to service multiple customers with a single set of resources. The use of multicast communication requires that part of the on-demand nature of the system be sacrificed to achieve scalability and cost-effectiveness. One drawback to using multicast communication is that it complicates the provision of interactive VCR-style functions. Interactivity can be provided by either increasing the complexity of the customer set-top box or by modifying the semantics of the interactive functions to make them easier to provide. We describe a framework and mechanisms by which such interactive functions can be incorporated into a multicast delivery VOD system. Through the use of simulation, we evaluate and compare the performance of a unicast VOD system and multicast VOD systems offering various levels of interactivity.

*A prototype Audio-on-Demand system incorporating the ideas in this paper and running in a Sun OS environment is available by contacting the authors.

1 Introduction

A Video-On-Demand (VOD) service offers customers a large movie library from which they can select a movie to watch at any time they desire. *Interactive* VOD allows customers to interact with a movie being shown using VCR-style functions such as pause, rewind and fast-forward. The major challenge in providing VOD is handling the potentially enormous demand for audio and video in a real-time network with real-time interaction[1]. In typical VOD system architectures, the approach is to service customers individually by allocating and dedicating a transmission channel and a set of server resources to each customer. The problem with this approach is that it leads to an expensive-to-operate, non-scalable system. Most VOD trials to-date have only been geared toward servicing a small population of customers (see for example [2]). In these systems, as the number of customer requests increases, the quality of service can only be maintained by significantly increasing server resources and network bandwidth.

One of the best ways to improve system performance is to satisfy customer requests using delayed server response and multicast communication. Advantages are gained in both the server and the network when several similar requests occur at nearly the same time. In the *server*, requested data need only be accessed once. In the *network*, multicast communication can be used to reduce required bandwidth. Customers can share a single movie stream resulting in reduced system cost per customer and improved system scalability. This sharing of system resources contradicts, to a certain extent, the need for individualized service in an interactive VOD system. As it turns out, providing interactive functions in a multicast VOD system typically involves either added complexity in the set-top box or a redefinition of the interactive functions (from the traditional VCR-style). It may also be necessary to sacrifice some of the on-demand nature of the system. Systems in which some interactivity and part of the on-demand nature are sacrificed to achieve cost-effectiveness or other objectives are sometimes referred to as *Near* VOD systems[3].

In this paper we explore the tradeoffs inherent in the use of multicast delivery in an interactive VOD system. Our contributions include the development of specific mechanisms to (1) provide interactivity based on two previously proposed types of VCR-style functions[1, 3]; and (2) improve a server's ability to satisfy interactive requests by reserving *emergency channels* for high load periods. We also evaluate and compare the performance of several VOD systems showing the improved scalability properties associated with using multicast delivery. We also show the tradeoffs and costs involved in the provision of interactive functions in such an environment. Finally, we discuss our efforts to develop a prototype for the purpose of validating the feasibility of concepts discussed in this paper.

The use of multicast communication for the provision of scalable information services has been the subject of some research in the past (see for example [4, 5, 6, 7]). The use of multicast delivery in the context of VOD systems has been the subject of more recent research. However, much of the research[1, 3] has not provided performance results based on either simulation or analysis. In addition to the work described in this paper (reported previously in conference publications on which this paper is based [8, 9]), related efforts include [10, 11]. In [10] queueing and batching of requests are used and policies for selecting which movies to serve are evaluated. In that system customers that wait too long may renege and one of the objectives in choosing a scheduling policy is to minimize reneging. In addition the authors refer to the implementation of interactive functions

(particularly the pause/resume function).[1]. The work reported in this paper differs in that we assume no queueing of requests. The use of fixed-length time slots allows customers to be informed immediately after a request is made if it has been successfully scheduled or blocked. Blocked requests leave the system. In addition we consider a much wider variety of interactive functions and approaches to providing them. In [11] sharing server resources is accomplished by allowing customers to "catch up" with other movie streams by adjusting the playout speed of certain movies. No interactive functions are considered.

The remainder of this paper is organized as follows: Section 2 describes a reference video-on-demand system architecture and discusses its components and section 3 explains the basic elements in providing a multicast VOD system. Section 4 contains our proposals for providing interactivity within the context of a multicast delivery VOD system and discusses the required functionality in the set-top box and the server for two forms of interactive functions. Section 5 contains the results of a simulation study that we conducted to compare the performance of multicast and unicast VOD systems and the performance of the forms of interactive functions we propose. Section 6 describes our effort to prototype a multicast delivery VOD system with interactivity. The paper is concluded in section 7.

2 Video-on-Demand System Architecture

The traditional commercial television cable network has a star-bus topology which supports a one-to-many broadcasting paradigm. Furthermore, the network supports the multiplexing of many channels simultaneously[12]. Since the signal received by the customer is a combination of many channels, the customer must use a tuner to select the desired channel. Even with vast improvements in technology, the architecture of the traditional television broadcast network should remain essentially the same[13]. The broadcast signal may eventually become a digital signal carried over fiber optic cable to the home, but the need to broadcast programs will remain. For this reason, we adopt the system model shown in Figure 1 as the reference architecture for the VOD systems discussed in this paper.

It should be emphasized that the mechanisms we discuss and analyze in this paper are actually implemented in the end-systems (set-top box and server) [2] and are independent of the actual transport network details. The discussion in this paper is also, therefore, applicable to systems where the network employs other than the circuit-switched, cable-TV model [3]. It is sufficient for our purposes that 1) the network connecting the server and the set-top boxes be multicast capable and provide a user interface allowing customers to join and leave multicast groups efficiently, 2) the network provide a reasonably fast channel for conveying control messages.

In addition to the network there are two other major components in the VOD system:

- **The Server**: The server is responsible for: (1) receiving, processing, and satisfying customer

[1] The work reported in [10] although sharing some of the concepts developed by us in [8, 9] was conducted independently.

[2] Because interactive functions are performed at the end-systems (see section 4), no stream playout modification is required to deal with them. This end-system approach, therefore, has the added advantage of reducing server load and avoiding the problems addressed in [14].

[3] For a comparison of the different possible network technologies see [15].

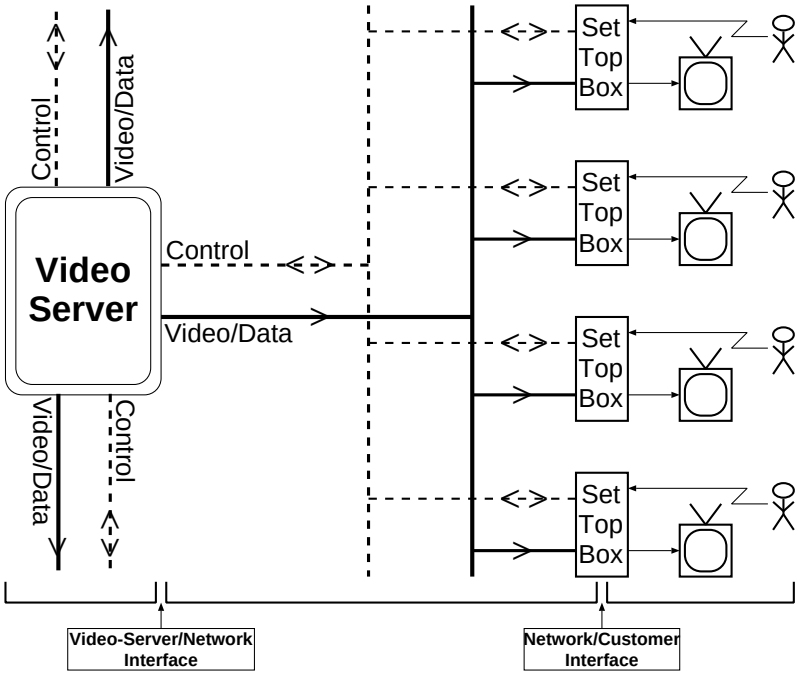

Figure 1: The architecture of our VOD system.

requests, and (2) establishing, maintaining, and modifying the movie streams used to deliver requested movies. Like the network, the server has a finite amount of resources. In order to accomplish the first responsibility, the server must be able to reserve resources and decide if a particular customer request can be satisfied given the current system load. Continuous and seamless playout is achieved by proper request management, and the ability of the server to retrieve data from a variety of storage media while still meeting real-time delivery deadlines. The issues involved in retrieving and sending frames have been the subject of some recent research (see for example [16]).

- **The Customer Interface**: The customer interface device, also called the *set-top box*, can take many different shapes and sizes, and perform a variety of functions. It is a simple channel tuner today, but may soon develop into a powerful processor rivaling the capability of personal computers. A non-traditional set-top box may actually be a personal computer sitting between the cable network and the television.

  Whatever the set-top box looks like, it must perform two functions: (1) sending, receiving and processing control messages and customer input; and (2) receiving, decoding and displaying frames. Satisfying (1) means the set-top box must act as a customer interface device accepting input from the customer and providing a way to select programs. If the VOD service includes interactive actions, the set-top box must handle these requests and accommodate any resulting changes in the movie playout.

  An important consideration in the evaluation of a VOD system is the cost of the set-top box. As we explore options for providing varying levels of customer interactivity, we will consider the operating complexity of the set-top box in the evaluation of the system.

Given these three components, we now describe the concept of a *logical channel* for the delivery of movies. In this paper, a logical channel, or simply a *channel*, is defined to be the set of resources in the server, network and set-top box necessary to provide continuous delivery of a movie to the customer. The key resources include, but are not necessarily limited to: (1) disk bandwidth in the server, (2) transmission bandwidth in the network, and (3) decoding hardware in the set-top box.[4] The size of systems discussed in this paper are measured in terms of the number of simultaneous channels that can be supported.

3 Multicast Delivery VOD – Basic Operation

In our multicast delivery VOD system, movies are made available only at the beginning of slots. The slot duration is on the order of minutes (in our analysis we use the range from 30 seconds to 20 minutes). A customer making a request will thus have to wait, on average, half a slot duration before the movie can start. For short slot durations (say 6 minutes) this should not affect the "on-demand" nature of the system.

When the server receives a customer request it determines if resources are available to service the request. The server uses information about outstanding requests and the availability of resources to accept or reject requests. Note that the server performs straightforward "First Come, First Serve" scheduling. Requests are not assigned priority, and no request is denied if resources exist to service it. Other scheduling disciplines are possible, and some have been investigated in [10]. Customers are informed through response messages whether their request is accepted or denied. All requests that arrive during the current slot are scheduled or rejected before the end of the slot. Figure 2 shows the flow of control messages between the server and set-top box.

The synchronization of movie start times to a slot boundary allows the server to serve all customers who request the same movie during a slot using a single channel. The server starts multicasting a movie at the end of a slot to all customers that request the movie during the slot. Note that we only start a movie stream if requests were made for it in the previous slot. The actual multicast groups are formed by having multiple set-top boxes listen to the same channel. Assuming a Frequency Division Multiplexed System, a multicast group is identified by a particular channel and joining a group can be accomplished by tuning to the appropriate frequency. We use the term "multicast communication" because even though in our reference architecture (Figure 1) every set-top box receives the transmission, not all customers have requested the stream. Only a subset of set-top boxes will receive and process a particular multicast movie stream.

Our ability to group multiple requests will depend on several factors. The longer the slot duration is, the more likely that several requests for the same movie will arrive during the slot. Of course, long slots detract from the "on-demand" nature of the system. The other factor is the popularity of a particular movie. The system is more likely to be able to group requests for currently popular movies than for a less frequently requested movie. Movie rental statistics[17] and Zipf[18] suggest that a small percentage of the movie offerings will experience the largest request volume. It is for these movies that the benefit of the use of multicast becomes apparent.

[4] A set-top box need only decode one stream at a time.

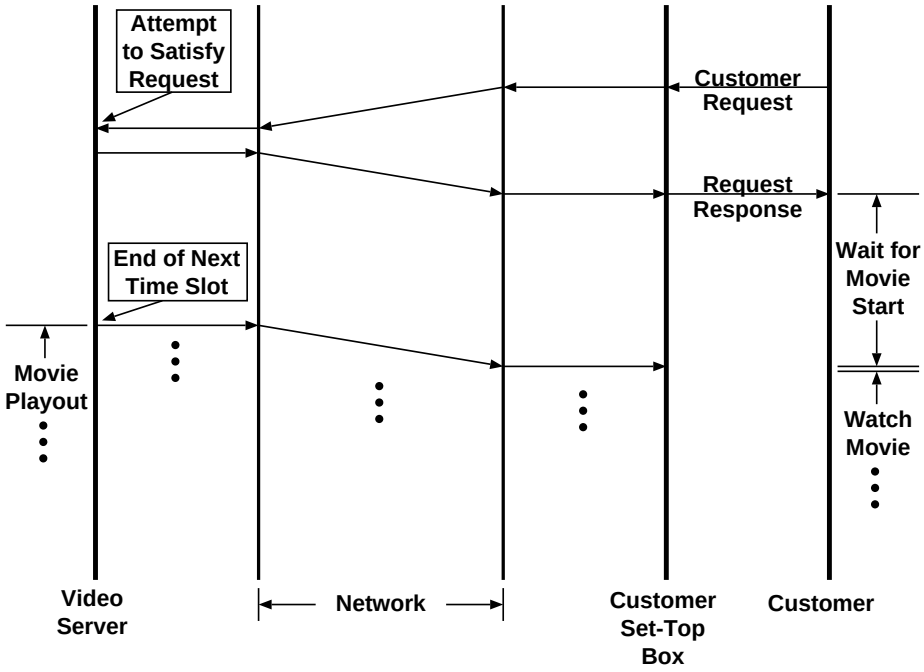

Figure 2: Operational overview of a multicast VOD system.

4 Interactive Functions in Multicast VOD

In this section we explore how interactive functions may be incorporated into a multicast VOD system. We first discuss some general aspects of interactivity in a VOD system. This is followed by the set-top box mechanisms used to support the interactive functions we propose. Finally we discuss the server functionality required to provide these functions.

4.1 Two Forms of Interactivity

In interactive VOD systems, a customer watching a movie will have the ability to control the playout of the movie. Customer interaction with on-demand movies can be similar to the interactivity customers have when they rent a movie and watch it using a video tape machine. In addition, the use of digital video will enable new paradigms for interactivity. In this paper, we limit our discussion to variations of the traditional VCR functions of pause, rewind, and fast-forward.

We distinguish between two different levels of interactive functions:

1. A VOD system offering *continuous* interactive functions allows a customer to fully control the duration of a pause, rewind, or fast-forward action. When a customer presses the rewind or fast-forward button, the movie starts playing at n times ($n > 1$)the normal playback rate in the reverse or forward direction. A pause action halts the movie playout until the customer resumes movie playback.

2. In a VOD system that offers *discontinuous* interactive functions, an action can only be specified for durations that are integer multiples of a predetermined *time increment*, τ. For

example, if this time increment is 2 minutes, a customer can only pause for durations that are $2, 4, 6, \ldots$ minutes long. For rewind and fast-forward, the customer can move the play-out point by periods of $2, 4, 6, \ldots$ minutes in the reverse and forward directions, respectively. Rewind and fast-forward actions are instantaneous.

The issue of which of these forms of interactivity is better is beyond the scope of this work. Experience with our prototype indicates that the discontinuous interactive functions are quite acceptable and sometimes preferable. As will be seen later, a system with discontinuous interactive functions requires less complexity in the set-top box.

4.2 Actual and Logical Start Times

To explain the effect of continuous or discontinuous interactive functions on the playout of a movie stream, we define *actual* and *logical* start times. The actual start time defines the time the movie started playing. The logical start time is defined as the current time minus the time it would have taken to watch the movie from its beginning until the point at which the customer is watching with no interruptions. Initially, the logical and actual start times are the same, but become different when interactive actions are initiated. For example, Figure 3 shows how the logical start time changes because of a pause action. The first graph in Figure 3 shows the effect of continuous pause actions of duration p_1 and p_2, and the second graph shows the effect of discontinuous actions of length 2τ and τ. Since continuous actions can be of any duration, there are no restrictions on the length of p_1 and p_2. However, discontinuous actions must be multiples of τ. Notice that when normal playout is occuring, the slope of the line is equal to one. During a pause, the slope becomes zero since real time continues, but the playout point is not advancing.

The two graphs in Figure 4 show how continuous and discontinuous rewind actions affect the logical start time. Since continuous rewind actions are at n times normal playback speed the slope during the rewind action is $-n$. For discontinuous rewind actions, the action is instantaneous and the slope of the line is infinite. A rewind action adds to the logical start time because a rewind means less of the movie has been shown and so the movie must have had a later logical start time.

Figure 5 shows the results of a fast-forward action. Any fast-forward causes a jump in the playout resulting in an earlier logical start time. The slope during a continuous fast forward action is $+n$ because the playback during a fast forward action is increased n times. For discontinuous fast forward actions, the change is instantaneous and so the slope is infinite. Again notice that discontinuous actions must be multiples of τ.

In a multicast delivery VOD system, more than one customer may be watching the same movie stream. It is, therefore, not always feasible to modify the movie stream as a result of an interactive action. Limited interactivity can be provided in the set-top box through the use of buffering. Additional interactivity can be provided by taking advantage of the fact that multiple streams for the same movie might have been started at other time slots. A customer that initiates an interactive action changes the logical start time of the movie he/she is watching. If local set-top box buffering cannot accommodate the interactive action, the customer is moved to an existing or new movie stream with an actual start time that matches the customer's logical start time. Parameters of the VOD system (such as slot length and amount of set-top buffering) need to be set such that the likelihood of this actual/logical start time match is high.

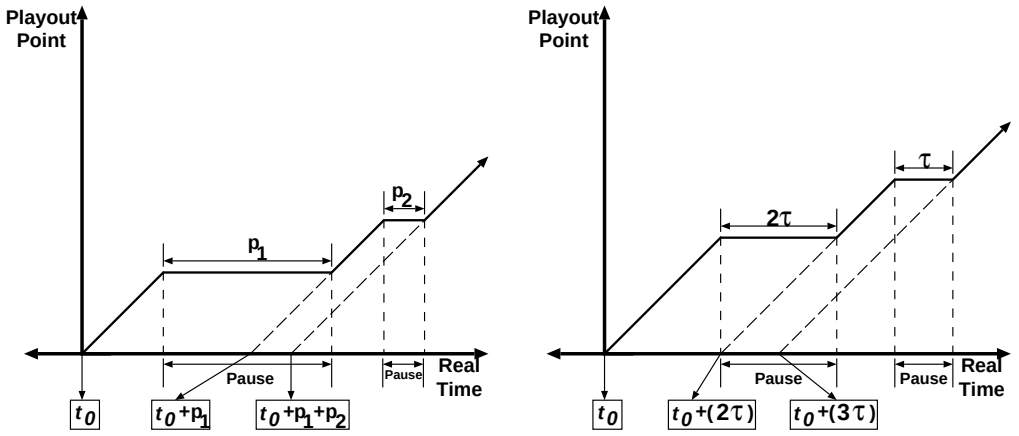

Figure 3: Comparison of the operation of the continuous and discontinuous pause functions.

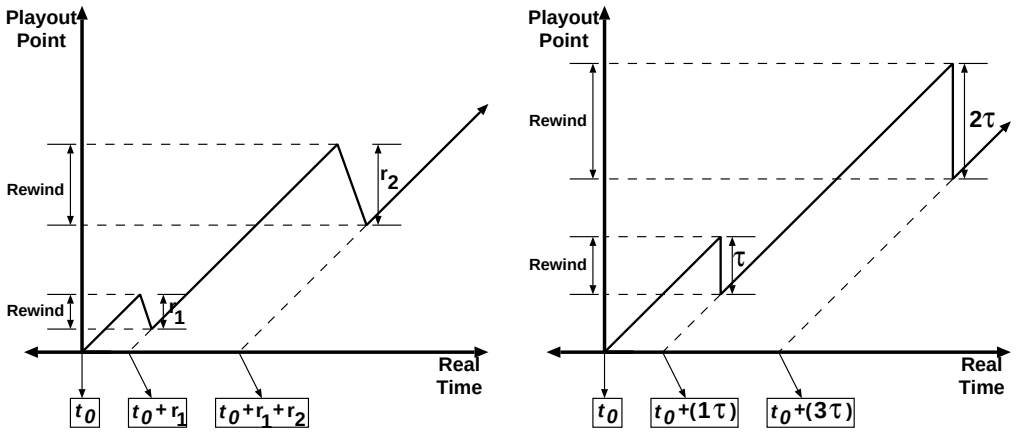

Figure 4: Comparison of the operation of the continuous and discontinuous rewind functions.

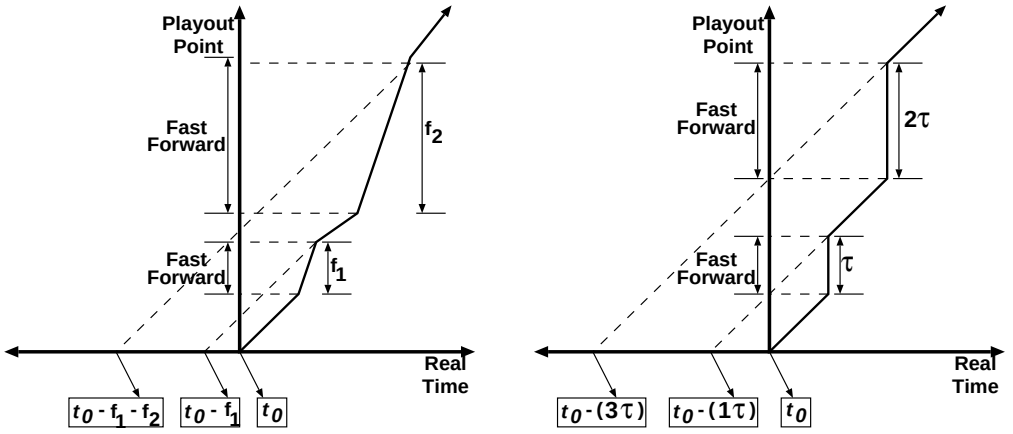

Figure 5: Comparison of the operation of the continuous and discontinuous fast-forward functions.

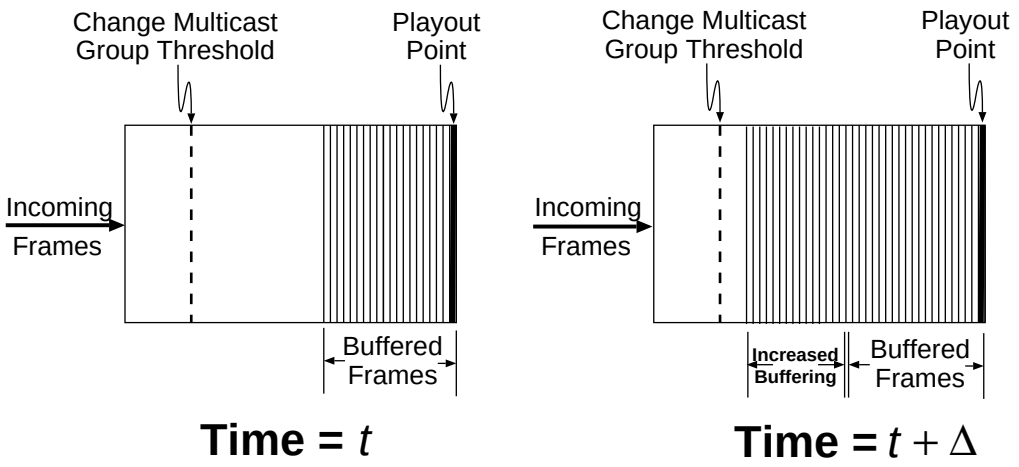

Figure 6: Typical state of the buffer in the set-top box.

4.3 Providing Continuous Interactive Functions

The operation of a multicast VOD system with continuous interactive functions is based on buffering in the set-top box and the ability of the server to move customers to different, existing movie streams with the appropriate start times.

4.3.1 Continuous Pause Capability

A small buffer is typically required in the set-top box to provide continuous playback. We augment this buffering to provide continuous pause capability. Figure 6 shows the typical state of the buffer which includes some number of frames that have not yet been played out.

The buffer acts as a first-in, first-out queue. As frames arrive, they are placed at the end of the queue. Playout of frames is from the beginning of the queue. When the customer is watching the movie, and not using the pause function, frames arrive at roughly the same rate as they are played out. When the customer initiates a pause action, the playout of frames is stopped, while new frames continue to arrive. The net effect on the buffer is that the number of frames stored increases. Figure 6 shows an example of how the state of the buffer changes during a pause in the movie.

This buffering mechanism is sufficient by itself to provide pause functionality in a multicast system as long as the total cumulative pause durations does not exceed the buffering capacity of the system. If a customer uses the pause function for more time than the size of the buffer allows, a change in multicast groups is required. For this to be possible we need to have the buffer be large enough to hold a slot's worth of frames.[5] When the number of frames in the buffer exceeds a pre-defined threshold, the set-top box sends a message to the server requesting that the customer be moved to the multicast group at the next later time slot. The detailed operation of the server is described in Section 4.5, but in terms of the set-top box operation, the server will respond either with a new channel or a request blocked message. If the set-top box receives a request blocked

[5] As an example, a 67.5 MByte buffer is needed to store 6 minutes of MPEG-1 compressed video at 1.5 Mbps.

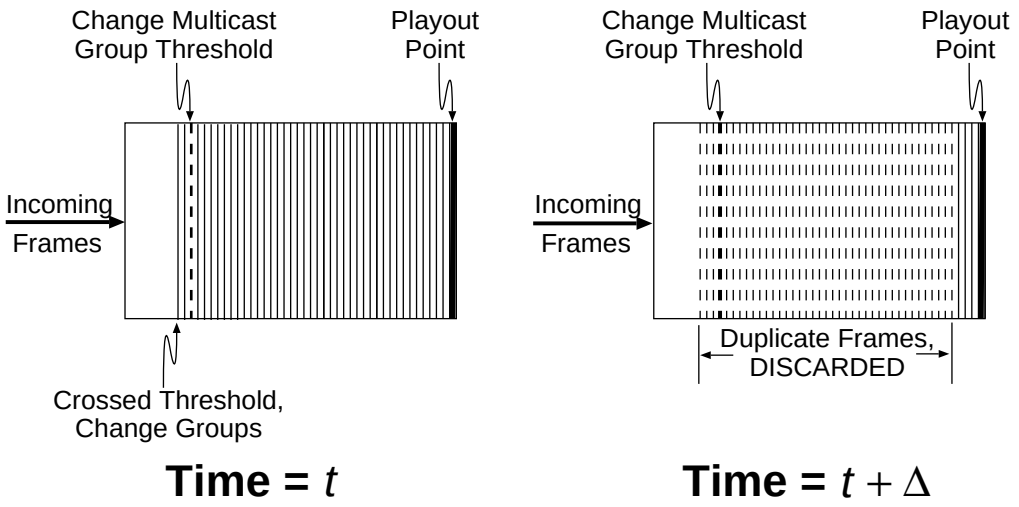

Figure 7: Effect of filling the set-top box buffer and the result of the multicast group change.

message, continued use of the pause function is blocked, and movie playout must re-start [6]. A blocked pause action represents a significant problem, and the server will attempt to minimize the probability of a blocked interactive action.

In the case of a successful multicast group change, the set-top box will tune to a new channel carrying the same movie, but with a later logical start time. Therefore, by tuning to the new channel, the customer will start receiving frames that are already stored in the buffer. The buffer can now be cleared up to the point where old frames are duplicated by frames being received on the new channel. The effect of changing multicast groups is that by switching to a later movie stream, the buffer can be emptied since the customer will again receive those exact same frames. Figure 7 shows what happens when the buffer breaks the threshold, and a multicast group change occurs.

If the customer continues to pause, and the buffer fills up again, the same process will be repeated. The customer will leave the group it just changed to, and join a new group with an even later logical start time. The buffer is designed to hold a time slot's worth of frames so that when the buffer fills up, the customer can be moved to the movie stream started at the end of the next time slot. Buffering in the set-top box allows the customer to pause up to the length of a time slot before requiring a multicast group change.

4.3.2 Continuous Rewind and Fast-Forward Capability

Using additional buffering, continuous rewind and fast-forward capability can be provided to a limited degree. As we will show, the availability of the rewind function is dependent on the buffer size; more buffering can be used to provide greater availability. For the fast-forward function, availability varies depending on the state of the buffer. Providing fast-forward in most cases will be extremely difficult.

[6] Alternatively, and as suggested in [10], the customer can be removed from the movie stream and allocated another channel when the resumption of the movie is desired. This may result in the customer having to wait until resources are available before the movie is resumed.

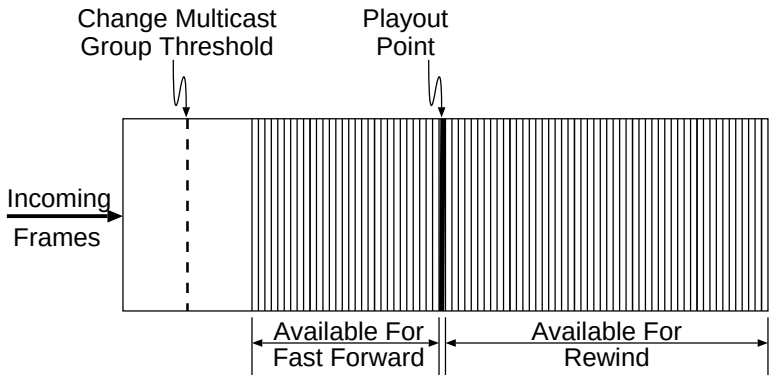

Figure 8: Typical state of a buffer in the set-top box.

Figure 8 shows an example of the state of a buffer capable of supporting pause, limited rewind, and an even more limited fast forward function. This buffer is the same as the pause-only buffer described in the last section except that additional buffering is logically attached to the end of the pause-only buffer. This additional buffering is reserved for the storage of frames that have already been displayed. These frames are used to provide limited rewind capability. The larger the rewind buffer, the more rewind capability that can be provided. The total size of the buffer is equal to the amount of buffering needed to store one time slot's worth of data, plus the rewind buffer, plus a small amount of buffering to provide continuous playout. The playout point will vary depending on the state of the buffer and a customer's use of the interactive functions. The set-top box will try to keep the playout point in the middle of the buffer so that: (1) there should always some frames available for rewind, and (2) there will be a time slot's worth of buffering available to store un-played frames.

During normal playout, frames will arrive and be buffered. If the buffer reserved for old frames is full then as frames arrive the oldest frames will be discarded. The location of the playout point in the buffer can be adjusted to the right or left by discarding the oldest frames at a rate slower or faster than the arrival rate of frames from the server. Decreasing or stopping the throw-away rate of the oldest frames increases the number of frames available for rewind. Temporarily increasing the throw-away rate increases the amount of buffering available to store incoming frames. When a customer initiates an interactive action, the state of the buffer changes as follows:

Pause: A pause action temporarily stops the movie playout. If the number of frames available for rewind is to be maintained, then the discarding of the oldest frames must also be stopped. In any case, frames will continue to arrive and must be buffered. If the buffer fills, a successful multicast group change will eliminate a time slot's worth of data.

Rewind: The first effect of a rewind action is to stop the discarding of the oldest frames. The second effect is that playout will reverse, and previously displayed frames will be re-displayed. A rewind action can continue as long as there are still old frames in the buffer which can be re-displayed. During a rewind action, frames from the server will continue to arrive and must be buffered. If the buffer fills during a rewind action, a time slot's worth of data can be discarded if a successful multicast group change is executed.

Fast-Forward: A fast-forward action can only be performed if there are frames buffered which

are waiting to be played out. If these frames exist, the playout rate can be increased by not displaying some of these frames. However, if there are only a few such frames then a fast-forward action can not be provided without requiring some special action on the part of the server.

Using buffering in the set-top box, a multicast VOD system can only provide intermittent continuous rewind and fast-forward functions. Any attempt to provide a more consistently available version of continuous rewind and fast-forward functions will require more than just buffering. For this reason, we conclude that among the continuous interactive functions in our multicast VOD system, only the pause function is realistically implementable with a low-complexity and low-cost set-top box. The difficulty in providing continuous functions should be contrasted with the ease with which discontinuous functions (as described next) can be provided.

4.4 Providing Discontinuous Interactive Functions

The operation of the set-top box in a VOD system with discontinuous interactive functions is much simpler than in a system with continuous interactive functions. By setting the time increment τ to be equal to the slot length, no set-top box buffering is required, and each interactive action requires a multicast group change. For discontinuous rewind and fast-forward, the set-top box sends a request to the server that includes the desired new logical start time. When a successful response is received it will indicate the appropriate channel to tune to, the set-top box simply tunes to the new channel.

For a discontinuous pause request, the operation is slightly different. When a customer starts a pause, a request is immediately sent to the server including the desired new logical start time. A pause is treated just like a rewind at the server, and it responds with a new multicast group channel. The set-top box immediately tunes to the new channel, and starts receiving frames that have already been displayed (just like a rewind). Instead of displaying these frames, they are discarded until the set-top box receives the frame that was being displayed when the pause action was started. When this frame is received, playout continues. Note that it takes the duration of the discontinuous pause to tune to the new channel; and then receive and discard previously displayed frames until the current frame is re-received.

4.5 Server Operation for Interactive Functions

When a customer issues a request for an interactive action, a message is sent to the server with the new logical start time. Such requests are treated by the server in effect as requests for a multicast group change. The server then determines the number of customers watching the same movie at the original logical start time and at the new logical start time. Using this information the server determines if the interactive action request can be granted. The procedure used by the server is as follows:

1. The server must first determine if the resources exist to grant the interactive action request. A request can be granted if one of the following two conditions can be met:
   (a) If another stream exists which is showing the same movie at the new logical start time.
   (b) If no such stream exists, but resources are available which can be allocated for the creation of the appropriate stream.

2. If the interactive action request can be satisfied, the customer who performed the interactive action is moved to the existing or newly created stream. Depending on the number of customers left in the original stream, one of two scenarios is possible:

   (a) If the customer who moved to the new stream was the only one watching the old stream, the stream is no longer needed and can be deallocated. These resources can be used to satisfy other initial requests or interactive action requests which require a new channel.

   (b) However, if there are other customers watching the stream, then no deallocation can occur.

Emergency Channels: It should be noted that there is always the danger that a request to change to a different movie stream cannot be granted. This will happen when there are no existing movie streams to satisfy the request, and no idle channels are available. We consider this type of interactive action blocking to be unacceptable and would therefore like to design a system where the probability of this blocking is extremely low. Our approach is to trade-off slightly higher initial movie request blocking with lower interactive action blocking. This is accomplished by reserving a number of *emergency channels*. These are not available to satisfy initial requests and are used exclusively to serve requests to change movie streams that would otherwise be blocked. In the extreme case when no emergency channels are available, interactive actions that cannot be accommodated are blocked. A blocked continuous interactive action means that the action is ended early and playout is resumed. A blocked discontinuous interactive action is not allowed to occur and playout continues as if no request was made.

5 Performance Evaluation

5.1 Simulation Model

In order to evaluate the performance of multicast delivery, we developed a simulation model and used it to generate results for each VOD system described in this paper. The model uses customer behavior information to simulate the operation of VOD systems. Performance is measured in terms of the server's ability to satisfy customer requests. The model is based on two assumptions:

1. The number of requests that arrive during the six evening hours known as "prime time" will be significantly higher than any other time of the day[3]. During this period, the scalability of a system becomes critical. For this reason, our simulation only measures performance during these six hours.

2. The request rate is measured in terms of the total number of requests that arrive during the prime time hours. Requests are uniformly distributed over this six hour period.

At the beginning of the simulation all channels are assumed to be free. At the end of the prime time period no more requests arrive, but the simulation continues to run until all active requests are completed. This allows a fair comparison to made among the performance of various systems.

5.1.1 Systems Under Consideration

The use of rewind and pause by the customer is essentially transparent to the server. From a system operation point-of-view, the server receives customer requests to either jump back to another group (in the case of fast-forward), or jump forward to another group (in the case of pause or rewind). From a performance standpoint, a fast-forward action will improve performance since it will take less time to play a movie as compared to a movie in which fast-forward is not used. The use of fast-forward means resources will be freed sooner.

Because of this and because of the similarity between pause and rewind actions, the worst case performance of a discontinuous VOD system can be approximated by only modeling pause requests. We, therefore, focus on systems with pause only interaction in our evaluation and consider the following three VOD systems:

1. **Unicast VOD with Continuous Pause**. This is a VOD system providing service using a single channel per customer. A continuous pause function is provided.
2. **Multicast VOD with Continuous Pause**. This is a multicast system that provides a continuous pause function, but at the expense of additional buffering in the set-top box.
3. **Multicast VOD with Discontinuous Pause**. This is a multicast system that provides a discontinuous pause function. This system does not require the large amount of buffering required in the continuous system.

5.1.2 System Parameters

The system parameters used to understand the performance of the VOD systems include the following:

1. **Number of Customers Making Requests**. This is the total number of customers who will make requests during the simulation period. Each customer makes exactly one request, and does not re-try if the request is denied.
2. **Total Number of Channels**. This is the number of movie streams which can be supported simultaneously.
3. **Number of Emergency Channels**. This is the number of channels reserved for preventing interactive actions from being blocked. These channels cannot be used to satisfy initial customer requests.
4. **Slot Length**. This defines the length of time between movie starts as well as the time increment for discontinuous actions.
5. **Number of Offered Movies**.
6. **Movie Length**. We assume all movies to be of the same length.

The rest of the system parameters are used to model the behavior of each customer. They include the following:

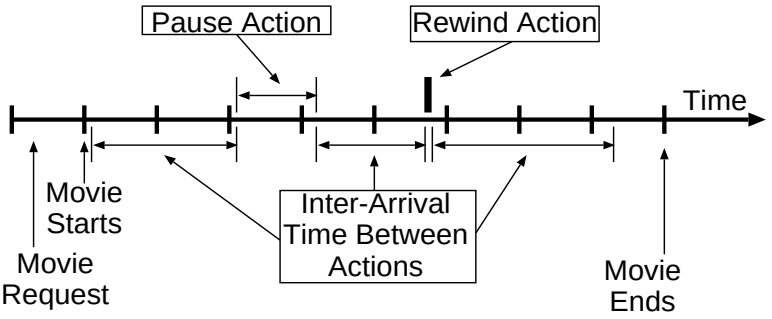

Figure 9: Time line showing movie playout and customer interactive actions.

1. **Time of Request**. A request is equally likely to be for any slot within the six hour prime time period.

2. **Movie Selection**. We assume the probability that a request for movie i is given by q_i. The q_i's are assumed to follow Zipf's Distribution[18] i.e., if $q_1 \geq \ldots \geq q_L$ where L is the number of movies, then $q_i = c/i$ where c is a normalizing constant such that $\sum_{i=1}^{L} c/i = 1$.

3. **Interactive Function Usage**. Figure 9 represents the viewing pattern for one customer watching one movie. After an initial request has been made, and the customer starts watching the movie, the customer's use of a interactive function can be modeled as an event arriving periodically, and then being processed. After an action is completed, the customer continues to watch the movie until the next action time. The duration of a pause is uniformly distributed between 60 and 600 seconds. For discontinuous interactive actions, the actual length of the pause is then rounded up to the next integer multiple of the time increment. The time between the conclusion of one pause action request and the start of the next is assumed to be exponentially distributed with a given average.

Table 1 shows a complete list of the parameters studied including the range of values tested and the nominal values.

5.1.3 Performance Measures

In evaluating and comparing the performance of the simulated VOD systems, we used the following two performance measures:

1. **Initial Request Blocking**. This is the percentage of initial requests that are blocked. A request is blocked when there are no free non-emergency channels to allocate to the new movie request.

2. **Interactive Action Blocking**. This is the percentage of customer pause attempts that were blocked because there are no free or emergency channels.

Factor	Range of Values Tested	Nominal Values
Number of Customer Movie Requests	2000 to 6000 requests	4000
Total Number of Channels	200 to 1400 channels	800
Emergency Channels	0 to 16 channels	4
Slot Length and Time Increment for Discontinuous VCR Actions	$\frac{1}{2}$ to 20 minutes	6
Average Time Between Customer VCR Actions	5 to 60 minutes	30
Number of Offered Movies	Fixed	500
Length of Movie	Fixed	120

Table 1: The range of values studied, and the nominal value for each factor.

5.2 Performance of Multicast VOD with Interactive Functions

Figure 10 shows the results of varying the number of customers making requests. The results show that as the number of requests increases, so does the initial request blocking probabilities in all three systems. In the unicast system, the blocking probability increases much more quickly. For the multicast systems, the blocking probabilities are similar with the discontinuous system having a slightly higher blocking probability. This is because the durations for discontinuous pause actions are computed by rounding the continuous pause action request lengths to the next higher time increment. Longer VCR actions mean resources must be held longer since the duration of the movie is increased. This in turn increases the initial request blocking probability slightly. The pause action blocking probability increases slightly for both multicast systems as the number of requests increases. The pause action blocking probability in the discontinuous system is slightly higher because all pause actions require a multicast group change. Because resources are allocated on a per customer basis with no sharing, there is no pause blocking in a unicast VOD system.

Figure 11 shows that as the number of total channels is increased, the blocking probabilities in all three systems decreases. Blocking in the multicast systems decreases more rapidly than in the unicast system because each multicast channel is capable of servicing multiple customers. The two multicast systems have similar initial request blocking probabilities. The pause action blocking probability in the discontinuous system starts out higher than in the continuous system. However, as the number of channels increases, both probabilities become almost zero. Note that it takes about 1000 channels (or a 4:1 ratio of customers to channels) to bring the multicast system's initial request blocking probability to 0. At this ratio the unicast system's blocking probability is still approximately 30%.

Figure 12 shows that as the number of emergency channels is increased the pause action blocking probabilities for the multicast systems decrease and become almost zero. The initial request blocking probabilities for the multicast systems increase slightly as more channels are reserved for

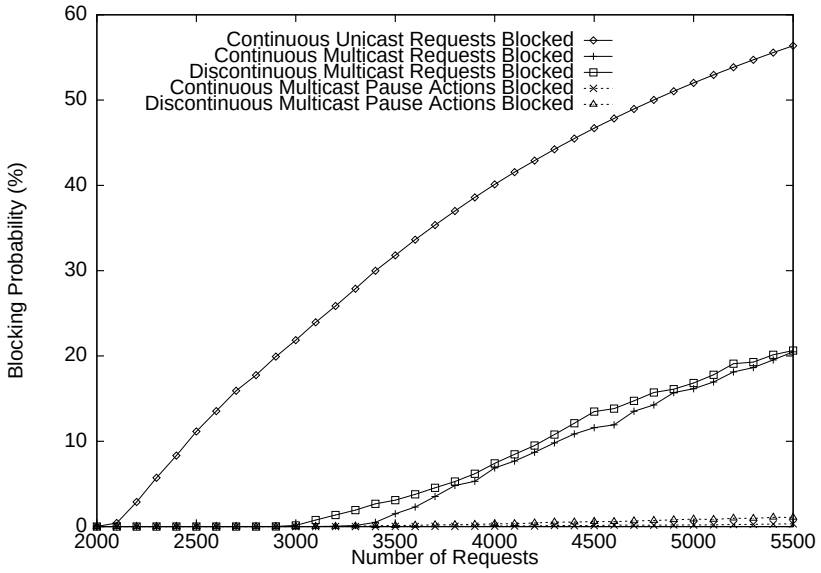

Figure 10: Blocking probabilities as a function of the number of customers making requests.

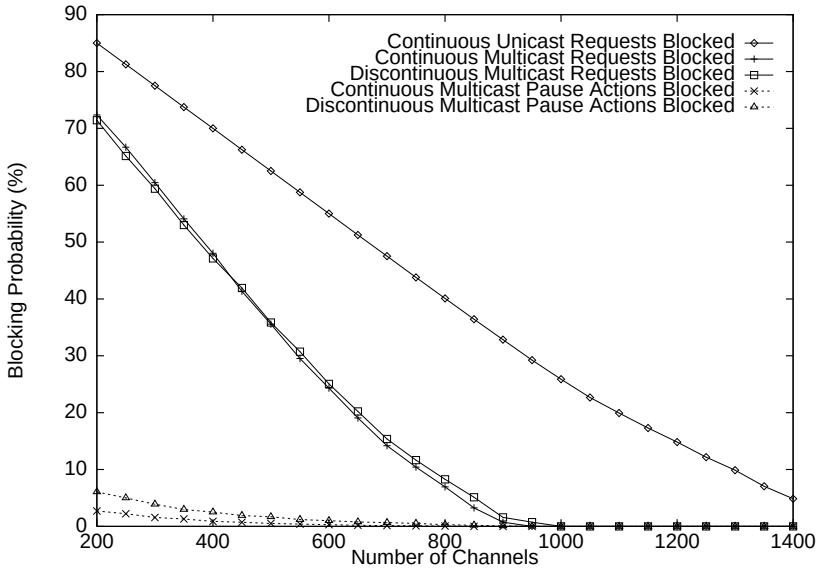

Figure 11: Blocking probabilities as a function of the total number of channels.

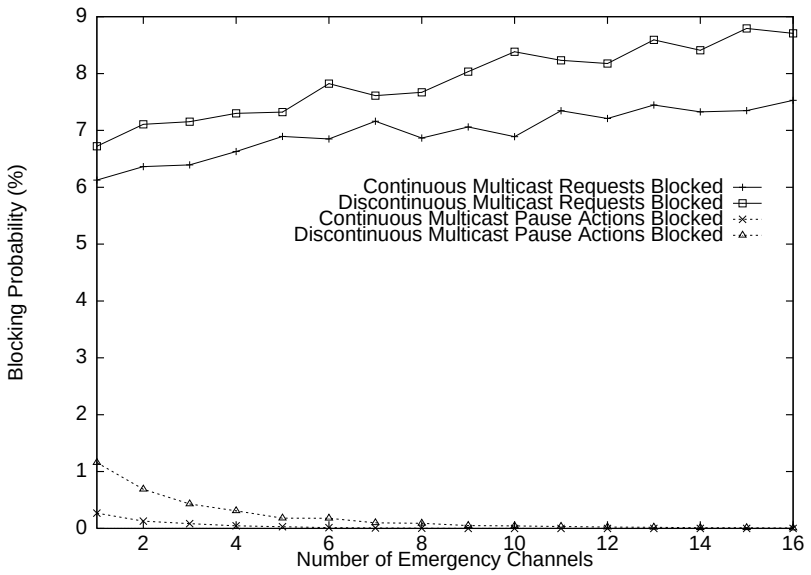

Figure 12: Blocking probabilities as a function of the number of emergency channels.

handling multicast group change requests. Even a small number of emergency channels (1% of the total number of channels) effectively reduce pause blocking to near 0%.

Figure 13 shows that as the slot length increases, the initial request blocking probabilities in both multicast systems decreases rapidly. Initially, very small slots approximate a unicast system since the probability of receiving multiple requests for the same movie in a short slot is very small. The initial request blocking probability in the discontinuous system is slightly higher because the length of discontinuous pause actions is slightly longer. The pause action blocking probability for both multicast systems remains consistently low.

Figure 14 shows that as the intensity of customer interactive actions decreases i.e., the time between the conclusion of one pause action and the start of the next increases, the initial request blocking probabilities for all three systems decreases. This interaction time is exponentially distributed with a mean time measured in minutes. For all three systems, frequent interactive actions mean that the playout time of a movie will be longer. In a unicast system, this means a customer's channel will be held for a longer period of time. In the multicast systems, more resources will be needed to establish movie streams at later time slots. Channels normally used to satisfy new customer requests will instead be used to satisfy frequent interactive actions. The pause blocking probability remains consistently low.

Figure 15 shows the number of channels needed to support different numbers of customer requests. The graph shows channel requirements for unicast and multicast systems with and without interactivity. Acceptable performance is defined to be an initial request blocking probability of less than 5%. One result is that unicast VOD systems require many more channels than systems using multicast delivery. As the number of requests increases, the rate at which channels must be added is higher in the unicast systems. For the nominal value of 4000 requests, unicast requires a system that is almost twice as large as a multicast system offering the same level of service.

Figure 15 also shows fewer channels are needed for systems which do not provide interactivity.

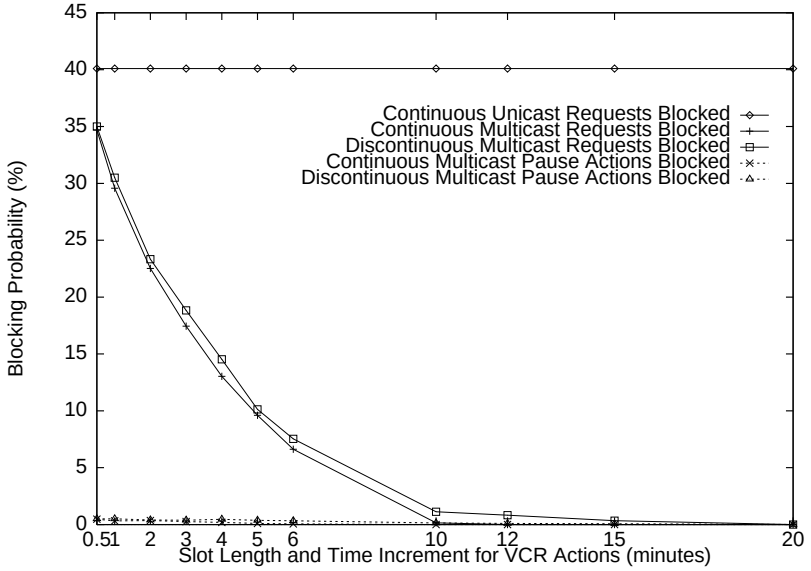

Figure 13: Blocking probabilities as a function of the slot length.

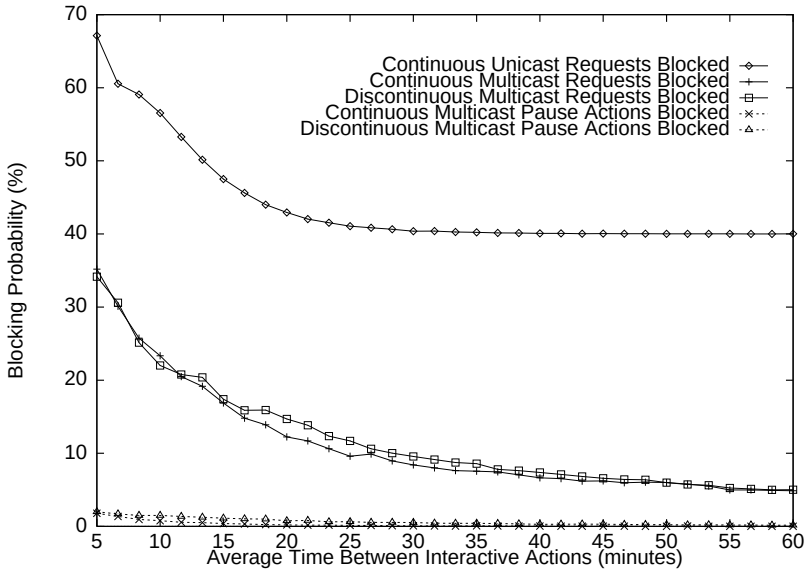

Figure 14: Blocking probabilities as a function of the intensity of customer pause actions.

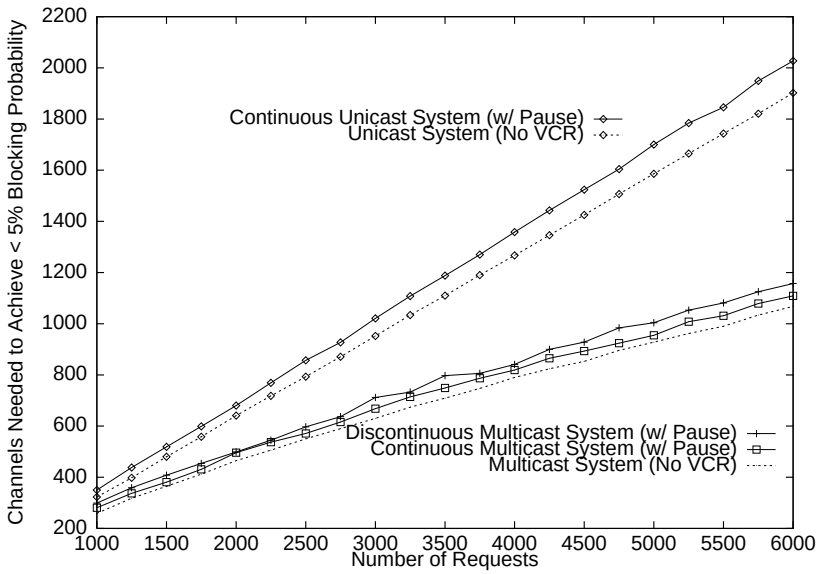

Figure 15: Required capacity for systems with initial request blocking $< 5\%$ as a function of the number of customer requests.

A unicast system or a discontinuous multicast system can provide interactivity with additional capacity of close to 8%. For the continuous multicast system, interactivity can be provided with additional capacity of just less than 5%. Less capacity is needed in the continuous multicast system because some of the VCR actions are handled in the set-top box through the use of buffering.

6 Prototyping Effort

An Audio-on-Demand (AOD) prototype was built to test the feasibility of the concepts proposed and simulated in this paper. Instead of providing movies, the customer is offered a library of songs. Audio has many of the same challenges as video, but the low/constant bit rate characteristics of μ-law encoding make an audio server easier to implement. Because songs last on the order of minutes, the prototype uses a shorter slot length and time increment for discontinuous VCR actions. Songs use a slot length of 10 seconds for an average song duration of 3.5 minutes as compared to the simulated movies which use a 6 minute slot length for movies 120 minutes long.

Building a large-scale server with hundreds of channels serving thousands of customers would be extremely difficult. Therefore, the primary purpose of the prototyping effort is to validate the feasibility of providing one-to-one interactivity in a multicast delivery system. The system design is capable of supporting a large number of channels, but actual available resources limited the prototype to a few channels. However, this is sufficient to demonstrate basic system operation and to experiment with different initial waiting times and types of VCR actions.

The delivery of multicast data over an IP-based internetwork is relatively straightforward[19]. When started, both the AOD server and clients open a UDP socket to send and receive control messages, i.e. initial song and group change requests/responses. When a channel is allocated, the

actual data transmission occurs using IP multicast and a unique UDP port number. A new UDP port is used each time a channel is allocated. The mechanism to switch between streams is simply a matter of closing the open socket and opening a socket for the new UDP port number. The only major difficulty is finding an efficient way to capture input on the control channel, data channel, and customer interface. The solution is to use alarms and asynchronous I/O to alert the client when data has arrived and action is required.

Providing VCR interactivity was more of a challenge. When a customer is to be moved to another group, a request is immediately sent to the server. The server uses information about the number of listeners to determine what action to take. The types of actions are the same as those discussed in Section 4.5, but instead of handling streams on channels, the server handles streams on multicast UDP ports. One major challenge is handling the *request delay*–the time between when a group change request is sent and when the customer hears the change. Because customers are simply moving among active streams (or newly created streams), there is no way to compensate for the processing delays associated with handling the request. The effect is a slight lengthening of fast forward actions, and a slight shortening of rewind actions. If group changes occured immediately, i.e. the request delay was zero, the actual duration of a discontinuous VCR action would be exactly equal to the slot length. For the prototype, the request delay was less than a second and was barely noticeable.

Finally, our AOD system uses discontinuous interactive functions. A rigorous evaluation of the human factors issues involved in the provision of this form of interaction is beyond the scope of this work. Our experience with the prototype, however, indicates that this form of interaction is quite acceptable. In certain ways, discontinuous interaction is more preferable than the traditional continuous VCR functions. Discontinuous functions have a "random access" feel and the change to a new playout point is immediate as opposed to the often time-consuming seek delays associated with continuous VCR functions.

7 Concluding Remarks

In this paper, we explored the benefits and tradeoffs associated with the use of multicast delivery in scalable VOD systems. One of the major challenges is the provision of interactive capabilities along with the use of multicast delivery. We first introduced a framework for incorporating interactive functions into a multicast VOD system. In particular, we presented mechanisms to handle continuous and discontinuous VCR-style interactive functions. Part of these mechanisms include reserving some number of emergency channels to provide continuous availability of interactive functions. Second, performance of VOD systems with and without interactivity using both unicast and multicast delivery was investigated. Through the use of simulation, we studied the effects of various parameters on the performance of these systems. A summary of the effects of these parameters is presented in Table 2. The results were also useful for evaluating the relative usefulness of multicast delivery and evaluating the cost of providing interactivity in VOD systems. Table 3 summarizes these comparisons. Finally we discussed our experiences in developing a prototype based on the concepts presented in this paper. Most surprising was our experience indicating customer acceptance of discontinuous VCR actions.

[1] Minimal buffering is required to handle jitter and provide continuous playout.

Factor	Influence on Performance
Number of Customer Movie Requests	More requests increases initial request blocking. With multicast, blocking increases more slowly because requests are grouped.
Total Number of Channels	More channels increases system capacity. Multicast increases capacity further because each channel can satisfy multiple requests.
Emergency Channels	More emergency channels means slightly higher initial request blocking, but much lower VCR blocking.
Slot Length and Time Increment for Discontinuous VCR Actions	Longer slot times reduces multicast blocking. More requests can be grouped during the slot. The tradeoff is a longer average initial wait and a longer discontinuous VCR action increment.
Average Time Between Customer VCR Actions	Pause and rewind actions increase the time that a channel is used, while fast-forward actions decrease the movie duration.

Table 2: A summary of the performance impacts caused by varying each factor.

System	Buffering in the Set-Top Box	On-Demand Nature Of Movies	Interactive Actions	Initial Request Blocking	Interactive Action Blocking	System Complexity
Unicast w/ Cont.	Not Required[1]	True	Continuous	High	Lowest (0%)	Low
Multicast w/ Cont.	Required	Near	Continuous (Limited Rew and FF)	Low	Lower	High
Multicast w/ Disc.	Not Required[1]	Near	Discontinuous	Low	Low	Moderate

Table 3: Summary comparison of three VOD systems.

References

[1] A. Gelman, H. Kobrinski, L. Smoot, and S. Weinstein, "A store-and-forward architecture for video-on-demand service," in *ICC '91*, 1991.

[2] J. Allen, B. Heltai, A. Koenig, D. Snow, and J. Watson, "VCTV: A video-on-demand market test," *AT&T Technical Journal*, pp. 7–14, Jan/Feb 1992.

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

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

[5] J. Wong and M. Ammar, "Response time performance of videotex systems," *IEEE Journal on Selected Areas in Communications*, pp. 1174–1180, Oct 1986.

[6] G. Herman, G. Gopal, K. Lee, and A. Weinrib, "The datacycle architecture for very high throughput database systems," in *Proceedings of SIGMOD*, ACM, 1987.

[7] D. Gifford, "Polychannel systems for mass digital communication," *Communications of the ACM*, vol. 33, pp. 141–151, February 1990.

[8] K. Almeroth and M. Ammar, "Providing a scalable, interactive video-on-demand service using multicast communication," in *ICCCN '94*, (San Francisco, CA), September 1994. file://ftp.cc.gatech.edu/pup/coc/tech_reports/1994/GIT-CC-94-36.ps.

[9] K. Almeroth and M. Ammar, "On the performance of a multicast delivery video-on-demand service with discontinuous VCR actions," in *ICC '95*, (Seattle, WA), June 1995. file://ftp.cc.gatech.edu/pup/coc/tech_reports/1994/GIT-CC-94-49.ps.

[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] L. Golubchik, J. Lui, , and R. Muntz, "Reducing I/O demand in video-on-demand storage servers," in *ACM SIGCOMM 95*, (Boston, MA), August 1995.

[12] K. Jacobsen and J. Cioffi, "High-performance multimedia transmission on the cable television network," in *ICC '94*, (New Orleans, LA), May 1994.

[13] R. Karpinski, "Fiber: Not just for telcos anymore," *Telephony: Transmission Special Supplement*, pp. 6–14, Dec 2 1991.

[14] 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), Oct 1994.

[15] R. Gusella and J. Limb, "Distribution of video to the home: Cell or circuit," Tech. Rep. HPL-94-07, HP Multimedia Technology Laboratory, March 1994.

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

[17] J. Yoshida, "The video-on-demand demand: Opportunities abound, as digital video becomes a reality," *Electronic Engineering Times*, March 15 1993.

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

[19] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, pp. 85–111, May 1990.