An Alternative Paradigm for Scalable On-Demand Applications: Evaluating and Deploying the Interactive Multimedia Jukebox

Kevin C. Almeroth

Department of Computer Science
University of California
Santa Barbara, California 93016-5110
almeroth@cs.ucsb.edu
(805)893-2777(office)
(404)893-8553(FAX)

Mostafa H. Ammar

Networking and Telecommunications Group
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
ammar@cc.gatech.edu
(404)894-3292(office)
(404)894-0272(FAX)

October 25, 1998

Abstract

Straightforward, one-way delivery of audio/video through television sets has existed for many decades. In the 1980s, new services like Pay-Per-View and Video-on-Demand were touted as the "killer applications" for interactive TV. However, the hype has quickly died away leaving only hard technical problems and costly systems. As an alternative, we propose a new *jukebox* paradigm offering flexibility in how programs are requested and scheduled for playout. The jukebox scheduling paradigm offers flexibility ranging from complete viewer control (true video-on-demand), to complete service provider control (traditional broadcast TV). In this paper, we first describe our proposed jukebox paradigm and relate it to other on-demand paradigms. We also describe several critical research issues including the one-to-many delivery of content, program scheduling policies, server location and the provision of advanced services like VCR-style interactivity and advanced reservations. In addition, we also present our implementation of a jukebox-based service called the *Interactive Multimedia Jukebox* (IMJ). The IMJ provides scheduling via the World Wide Web (WWW) and content delivery via the Multicast Backbone (MBone). For the IMJ, we present usage statistics collected over the last couple of years. Furthermore, using this data and a simulation environment, we show that jukebox systems have the potential to scale to very large numbers of viewers.

1 Introduction

Straightforward, one-way delivery of video programming through television sets has existed for many decades. In the 1980s, new services like Pay-Per-View (PPV) and Video-on-Demand (VoD) were touted as the "killer application" for next-generation Internet and TV services. However, the hype has quickly died away leaving only hard technical problems and potentially very costly systems. Even though VoD has been shown to be technically feasible, service providers have been hesitant to make the investment necessary for wide-scale deployment. Furthermore, almost all of the trials to date suggest VoD is too expensive and there is too little demand. What is needed, and what we propose, is a new paradigm offering flexibility in how programs are requested and scheduled for playout, ranging from complete viewer control (true VoD), to complete service provider control (traditional broadcast or cable TV). Furthermore, our proposed paradigm functions independent of a network's topology. Both a cable-TV- and Internet-based jukebox service are possible. And with solutions to the key problems facing each network–high quality delivery in the Internet, and bi-direction communication in cable TV systems–the jukebox paradigm could be developed into an attractive commercial service.

The jukebox paradigm we propose is based on the premise of allowing any viewer to watch any other viewer's requested program. Program requests are scheduled on one of a system's channels using a set of scheduling policies. The schedule is then made known to all potential viewers. Any viewer who wants to watch a program on a particular channel simply "tunes" to that channel. Content on each channel is delivered from a server to all viewers watching that particular channel using either a one-to-many (multicast) or a one-to-all (broadcast) communication facility.

In this paper, we describe the jukebox paradigm and relate it to other on-demand paradigms. We also describe some of the challenges in providing an on-demand program service. Of particular interest are issues like the best way to handle viewer requests, how to provide VCR-style interactive functions, and how to track viewer usage patterns. We also describe our efforts to prototype a jukebox-based service. The *Interactive Multimedia Jukebox* (IMJ) provides scheduling via the World Wide Web (WWW) and content delivery via the Multicast Backbone (MBone). We discuss some of the issues related to building the IMJ and our ongoing efforts in improving the jukebox paradigm. Part of these efforts have been driven by the collection of user feedback data collected over the past couple of years. We have been able to collect data about (1) the number of hits to the IMJ WWW page, (2) the program requests made, and (3) the number and duration of channel viewers. Finally, while prototype usage gives us valuable feedback, it does not give insight into how well jukebox systems operate for much larger audiences and much higher loads. To this end, we have created a jukebox simulation environment and evaluate the performance of larger jukebox systems and varied request scheduling policies.

The jukebox paradigm is an effort to bring together work in several research areas. One area is the scalable delivery of video-on-demand (VoD) service using multicast communication. True VoD, in which all viewers get their own resources, requires very large systems to provide adequate performance. These systems are expensive and provide few revenue opportunities for service providers. One solution is to batch multiple requests for the same program into a group and then service them using one audio/video stream multicast to all group members[1, 2]. This solution was mainly proposed for cable-TV based infrastructures which provide many channels, a large numbers of customers, and broadcast-only communication. Within the Internet, the MBone[3] has been the focal point for developing multicast[4] and real-time protocols[5] for the scalable delivery of multimedia streams. Like the IMJ, related efforts are looking at extending the use of the MBone beyond applications like interactive conferencing and program broadcasts[6, 7, 8, 9]. An inherent need when transmitting reasonably high bandwidth media streams is the need to deal with receivers who have heterogeneous bandwidth capabilities. Researchers are beginning to experiment with layered encoding or multi-stream schemes[10, 11, 12, 13]. Recent work has also looked at integrating the services of the MBone and the Real-Time Protocol (RTP) into the the WWW. Several researchers are looking at various ways of using the MBone and multicast protocols to deliver WWW pages[14, 15, 16, 17]. Other issues are based on the integration of WWW- and MBone-style conferencing[18, 19].

This paper is organized as follows. Section 2 describes our proposed jukebox paradigm and related paradigms. Section 3 details our implementation of a jukebox prototype called the *Interactive Multimedia Jukebox* (IMJ) and presents analysis and conclusions based on usage data. Section 4 discusses the results of a simulator used to evaluate the performance of large-scale jukebox systems.

Section 5 lists a set of advanced services that might be offered in a jukebox system. The paper is concluded in Section 6.

2 Program Scheduling Paradigms

2.1 Background

Our jukebox paradigm is based on a hybrid of program scheduling paradigms ranging from newer proposals like *true* VoD and *near* VoD to more traditional services like PPV and broadcast television[1, 2, 20, 21]. Traditional television is based on the premise of delivering as many channels of programming as possible. Viewer choice is the ability to switch between any of the available channels. Affecting what is shown on a particular channel is a slow process of feedback through program "ratings" followed by scheduling and programming changes based on evaluation of the ratings data by the broadcasters. PPV has attempted to give viewers additional choices, but fundamentally, the broadcasters still decide scheduling and timing. For a variety of reasons, PPV has never met the lofty financial goals set by many service providers.

Inherent in a discussion of video service paradigms are comparisons based on two factors. The first factor is the number of viewers who can watch a particular program stream. The second factor is a combination of how much viewer input is considered in program scheduling and whether the program schedule is developed in real-time or pre-arranged. Figure 1 shows the relationship between the paradigms mentioned so far. Existing television services like broadcast TV and PPV are located in the lower right of the graph. Users must plan their television watching habits around a pre-arranged schedule of programs. While this paradigm is quite limited, it has been the accepted practice since televison's inception.

At the other extreme, true VoD has the highest degree of viewer scheduling control. Program playout is based on specific viewer requests and playout starts as soon as the request can be satisfied. However, there will only be a single viewer per program stream, and so resources necessary to store, load, and transmit the program are allocated to a single viewer. Given that the standard for audio and video compression is likely to be MPEG-2, the storage and bandwidth requirements could

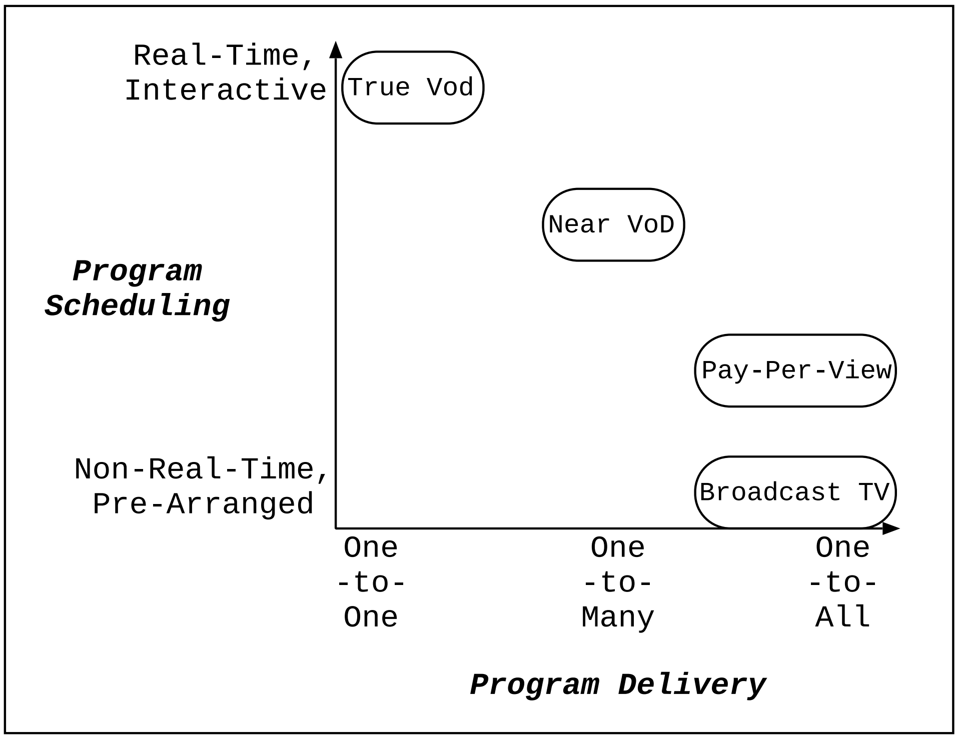

Figure 1: The relative position of scheduling paradigms.

easily exceed several Gigabytes for a two hour movie. Trying to make a profit will be very difficult especially since movies can typically be rented for only a few dollars.

With near VoD, the program start time is no longer immediate and some artificial delay is added. The hope is that multiple requests for the same program will be made in a short period of time. These requests are then *batched* and a single program stream is used to service the entire group. The assumption is that the network has an efficient multicast delivery facility and can provide a single program stream to several viewers. Near VoD has proven to be a scalable alternative to true VoD[1, 2]. The biggest limitation of near VoD is that it requires large viewer populations to achieve sufficient economies of scale. Furthermore, there is a tradeoff between scalability and the amount of time viewers must wait before a program starts.

2.2 The Jukebox Paradigm

The key design principal behind the jukebox paradigm is flexible scheduling based on a finite set of channels available to all viewers. The jukebox paradigm is designed to be scalable while offering flexibility in the way viewer requests are handled. The paradigm is based on three properties:

1. A set of "channels" are multicast to all viewers "tuned" to the respective channel.
2. Viewers may watch a program playing on any channel or make a request for something of their own choosing. Viewers' requests are scheduled on one of the jukebox's channels using some scheduling criteria.
3. A schedule of currently playing and scheduled programs, updated in real-time, is available to all viewers. Viewers can watch any program, including those scheduled by others, by tuning to the appropriate channel.

The jukebox paradigm is based on the operation of a music jukebox. Everyone in a room can hear what is being played on a music jukebox and song requests are made by individuals. These requests are queued, and then played in the order they are made. Anyone can make requests but everyone will hear what is played. Our jukebox paradigm offers some additional advantages. First, there can be multiple, distinct channels which means more choices for those who are just surfing. Second, the jukebox paradigm provides a visual interface about what programs are playing and scheduled. This provides an opportunity for viewers to have their decision influenced by what is already scheduled. Put another way, how many movie rental store customers know what they want to rent before entering the store? Most are influenced by the list of available titles or the suggestions of other customers or store employees. Viewers may scan the jukebox schedule and see something interesting which has only just started or will be starting soon. Third, there is opportunity to implement better scheduling policies than simple first-come, first-served.

One of the key features of the jukebox paradigm is that it is scalable while still providing a great deal of viewer choice. In the context of VoD systems, the term *scalable* means a system has the ability to provide service to additional viewers for a diminishing marginal cost. Using this definition, true VoD systems are unscalable because each additional viewer requires roughly the same set of resources required to deliver a program stream as was required by the previous viewer[1, 2]. On the other hand, near VoD is scalable because additional viewers can be accommodated by batching them with others who make the same program requests. One major disadvantage of near VoD is that there is still a relationship between the number of channels and the number of viewers. True bandwidth-limited systems may not be able to provide the additional channels needed to meet increased customer demand. Service will begin to degrade and customers will look elsewhere. The jukebox paradigm provides scalability using a fixed number of channels. Additional viewers can watch any channel for only the cost of joining the multicast group for that particular channel. The tradeoff with the jukebox paradigm is that as more viewers make requests the wait time for an individual viewer's request may increase. However, instead of being unable to watch anything, which occurs when a request is blocked in a VoD system, a viewer may be satisfied with something that is already playing or scheduled to start soon. The worst case occurs when there is a long wait time for a viewer's request and there is nothing in the schedule that interests the viewer.

In the jukebox paradigm, the ratio between the number of viewers and the number of channels is an important one. This ratio defines the type of service that viewers can expect. For example, at one extreme, if there is a large number of viewers and only a single channel, there is little chance that a viewer will make a request and then get to watch their program within a short period of time. At the other extreme, there will be as many channels as viewers making requests. In essence, each viewer will have a channel, like in a true VoD system. Furthermore, while there are economies of scale with the jukebox paradigm, they are not as severe as with near VoD. Jukebox systems can be successful offering only a few channels or many channels. As more subscribers join a service, additional channels can be added.

Another advantage of the jukebox paradigm is its flexibility in request scheduling. A great deal of flexibility can be provided because almost any set of policies can be implemented. Some of the possible scheduling policies include the following:

Shortest Wait Scheduling. The simplest policy for a multi-channel system is to schedule a viewer's request on the channel with the shortest wait time.

Content-Based Scheduling. Limitations may be imposed based on the content. For example, adult programming may be limited to certain channels or may only be played at certain times of day. New releases may be restricted to a certain set of premium channels. Also, specific channels may have a "theme". There could be a classic movie channel, sports channel, comedy channel, and/or cartoon channel.

Service Provider Scheduling. A service provider may have a desire or obligation to schedule certain programs at certain times. For example, a slot for the evening news may be scheduled each day at the same time. Service providers have an entire spectrum of control. At one extreme, the service provider does all the scheduling and the service becomes traditional TV. At the other extreme, the system is completely demand driven. Somewhere in the middle a service provider might reserve specific times slots on specific channels, or might use a combined system of some on-demand channels, and some broadcast channels.

Vote-Based Scheduling. A service provider may want to blunt the ability of individual viewers to control what programs are playing. For example, during peak periods, a service provider may impose limits on the number of programs a single viewer can request. A service provider may also institute a voting procedure. Only when enough votes from enough different viewers are received is a program actually scheduled. Viewers typically dislike having their "rights" restricted, but service providers must walk the fine line between making money and keeping customers happy. A side effect of this policy is that more popular programs are played more frequently, and in the extreme case, some very unpopular programs may not reach the "vote threshold" and would never be scheduled for playout.

Alternate Choice-Based Voting. One of the key advantages of a jukebox system is that requests are "shared" and any viewer can watch any channel. As a result, if a system is heavily loaded and a viewer's request is scheduled at the end of a long queue, the expected wait time may be unsatisfactory. As an alternative, the viewer might have a second choice that is already queued and about to start. While the viewer might not derive as much "satisfaction" from watching a second choice, the shortened wait time might offset any marginal dissatisfaction. For intuition on this type of scheduling consider how people behave in a movie rental store. If their first choice is not in they will usually pick a second or third choice because they'd rather watch a movie that night than wait until a later night when their first choice will be available.

With all of these scheduling policies we make an implicit decision about scheduling multiple instances of the same program. If a requested program is already scheduled on the jukebox and has not yet started playing, then the latest request is ignored. As an alternative in a real system, the user can be told the program is already scheduled and given information about the channel and start time. If a request is made for a program that is actively being played on some channel then the program is allowed to be scheduled, subject to the limitations of other scheduling policies. The intuition is that users will not want to watch a program already started, but will be satisfied if the program has already been scheduled and is waiting to start.

Having described the jukebox paradigm in detail, we now re-examine the graph in Figure 1. Figure 2 shows the graph with the addition of the jukebox paradigm. Because of its flexibility, the jukebox paradigm extends over a large region. All of the paradigms described in Section 2.1 can theoretically be implemented using variations of the jukebox paradigm.

2.3 Architecture for a Jukebox System

A generic architecture for a jukebox system has four main components. Figure 3 shows the relationship between each of the components, and a description of each follows.

- **Scheduling Control and Schedule Display**: The scheduler receivers viewer requests, performs scheduling, controls the video server, and provides a schedule of programs to all viewers.

- **Video Server**: The video server transmits audio/video streams into the network. Programs are stored in compressed format, ready for transmission. The video server does not require significant CPU resources, but a system providing a large number of high quality channels may require fast disk access techniques[22].

- **Network**: The network must provide an efficient multicast facility and have sufficient bandwidth to meet viewer quality expectations. The network must also have bi-directional capa-

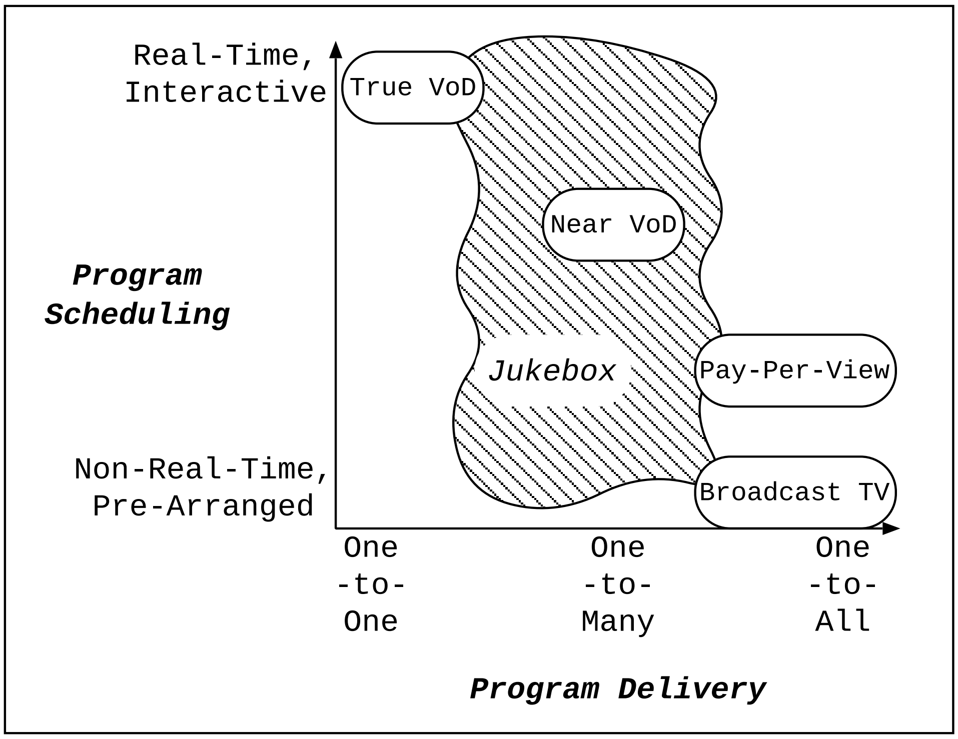

Figure 2: The relative position of scheduling paradigms including the jukebox paradigm.

bility in order to receive viewer requests and feedback. While this second requirement may seem obvious, it is worth mentioning in the face of today's asymmetric networks.

- **Receivers**: Receivers must be able to receive, decode, and display an audio/video stream. In addition, receivers must also be able to make program requests, and browse a program schedule. One of the keys to most streaming audio/video applications, especially when designed for entertainment, is that they should not require high end workstations.

At this point, it is worth mentioning how system capacity is measured. A *channel* is defined to be the set of resources in the servers and network necessary to provide continuous delivery of a program to all viewers. For certain topologies, like cable TV, a channel is an easily defined abstraction. In any case, jukebox systems will have capacity which can be measured in terms of the number of simultaneous logical channels that can be supported.

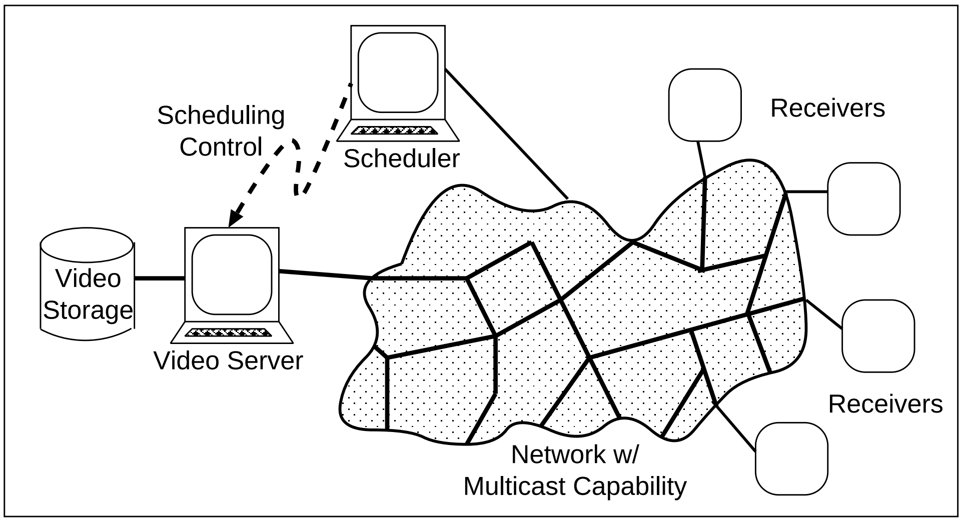

Figure 3: Generic architecture for a jukebox system.

3 Jukebox System Prototype

3.1 Prototype Details

The jukebox paradigm has been implemented, and the prototype is called the *Interactive Multimedia Jukebox (IMJ).* The IMJ uses the WWW for scheduling and program information and the MBone for multicast delivery of programs. By going to the IMJ home page (located at http://imj.gatech.edu), a viewer can see how many channels are available on the jukebox, what programs are currently scheduled for playout including start and end times, and what programs are in the jukebox library. Content being played on each channel is transmitted to all group members. Figure 4 shows the top of the IMJ home page including a snapshot of a sample real-time schedule.

Figure 4 shows that the set of IMJ scheduling policies is very simple. The time-to-live (TTL) value in IP packets is used to limit the scope of transmission. The TTL for channels 1 and 2 is set to 127 which means anyone in the world can receive these sessions. The TTL for the "GT Only" session is set to 15 which limits transmissions to the Georgia Tech campus. Only two global channels are provided so as not to put an undue bandwidth load on the world-wide MBone.

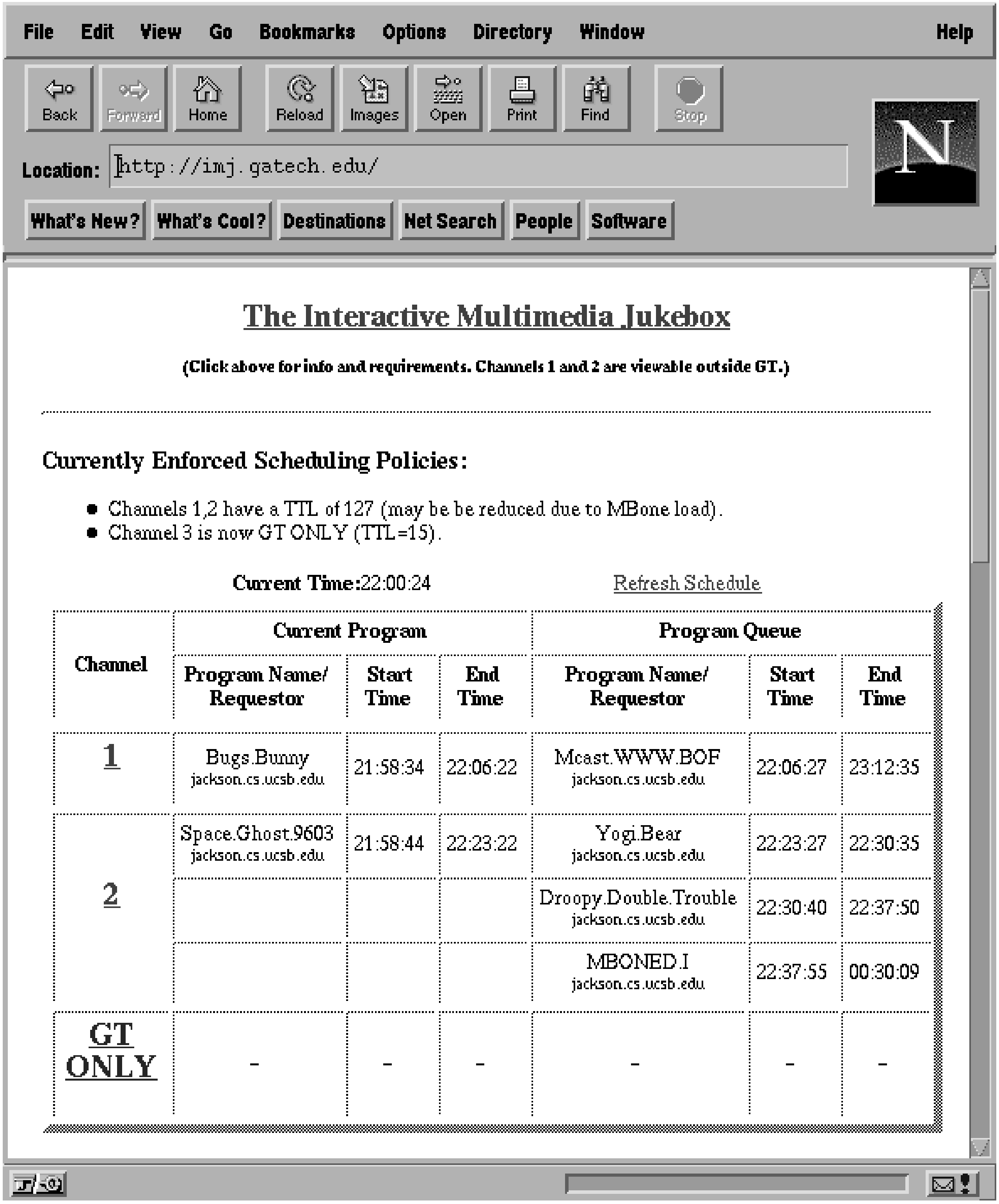

Figure 4: Snapshot of the IMJ scheduling page.

With the WWW providing the interface, the MBone provides the multicast delivery service over the Internet and the MBone audio and video tools provide delivery, decoding and display functions. Before being made available on the IMJ, content is encoded as an RTP packet stream using the rtpdump utility[23]. Quality levels for both audio and video are set at typical MBone session levels. Audio is encoded at roughly 39 Kbps using the Intel DVI audio format. Video is encoded at a constant bit rate of 128 Kbps using the H.261 coding standard. The IMJ library currently has more than 100 hours of programming. Plans to increase the library size are in the works and depend on the availability of new sources.

The actual architecture of the IMJ is very similar to what is shown in Figure 3. There is an HTTP server on one machine which serves requests for the IMJ home page and accepts program requests. Program requests are processed using a Perl script which passes the program name and information about the request to the scheduling daemon via a standard UNIX pipe. The scheduling daemon, written in C, processes and schedules the requests, and then updates the IMJ home page and its schedule in real-time by modifying the WWW page. Receivers are expected to periodically update their page by reloading the page from the IMJ WWW server. Embedded HTML flags automatically reload the page every 5 minutes. When it is time to start a particular program, the scheduler uses a remote shell command to the video server to start the audio and video streams via the *rtpplay* utility[23]. Stream synchronization is provided via RTP[5]. Because there are only two channels and per-stream bandwidth is relatively low, there is no need for specialized server hardware. Programs are stored as standard UNIX files and accessed from a disk local to the server via NFS.

3.2 Content in the IMJ Library

One challenge to making the IMJ a success was our ability to make interesting content available. To date, the three groups that have provided content for the IMJ are Turner Broadcasting Systems, Inc. (TBS), the Internet Engineering Task Force (IETF), and several groups within Georgia Tech. The ability to provide a useful service makes the IMJ more than just a prototype. In the case of TBS, characteristics of their cartoons have given us insight on how to provide entertainment-style content. Not only has the Turner content been the most requested, but the short cartoons seem to be of ideal length for the jukebox and their simple video images and lack of true voice synchronization help to hide some of the encoding artifacts. Our current agreement allows us to transmit content at a quality-limited rate of 128 Kbps. The main reason, which is mostly precautionary, is that lower quality content is less likely to be digitally recorded and does not detract from TBS's "real" TV channels. Readers should bear in mind that licensing content for Internet transmission is a difficult problem and content providers are justifiably hesitant. Table 1 is a list of content currently available on the IMJ and also lists the number of program requests received for each category. Because content constantly changes, the numbers in Table 1 only really represent the latest offering. Some programs, like the 41st IETF, have only recently been made available, and the total number of requests only covers the last couple of months. Regardless, the Turner cartoons and programs have proven to be the most popular.

Program Category	Number of Selections	Average Length Per Program	Category Library Size	Program Requests 1/1/97 to 5/27/98
Turner	25	30.6 min	7.39 hrs	6093
Georgia Tech	9	6.20 min	0.93 hrs	1185
38th IETF	10	1.50 hrs	15.0 hrs	640
40th IETF	29	1.61 hrs	46.9 hrs	306
41st IETF	20	1.81 hrs	36.2 hrs	205
TOTAL	93	N/A	106.42 hrs	8429

Table 1: Breakdown of IMJ library by program content.

3.3 Tracking Usage in the IMJ

Examining how the IMJ is being used is critical to understanding many aspects of the system. From a research point-of-view, we can find and correct problems and learn about behavior in systems using the new jukebox paradigm. From a pay-for-service point-of-view, tracking usage enables a service provider to decide when new programs should be added and when old programs are no longer worth offering. Furthermore, if a jukebox system is offered as an alternative to TV, service providers would like as much information about viewing habits as possible. Considering the importance broadcasters put on ratings, such data for the IMJ could be very valuable. For the IMJ we have the ability to collect data from three sources:

- **Program Requests**: Data about who, when, and what programs are requested. This data is collected by the scheduler and archived in a log file.

- **Jukebox Schedule**: Data about viewers who may not necessarily request a program but who check to see what has been scheduled. The IMJ WWW server access logs can be used for this information. Furthermore, because the IMJ page re-loads itself every 5 minutes we can monitor those viewers who point their browsers at the IMJ page for extended periods of time.

- **Program Viewers**: Because the IMJ uses the MBone tools for audio/video delivery, information can be collected about how many people are watching each session and for how long. This information is collected using a tool designed and developed by the authors called *Mlisten*[24]. In the past, it has been used to monitor multicast group behavior in traditional MBone sessions.

The IMJ has been in continuous operation as an MBone service since late in 1996. The results presented in the remainder of this section are based on operational data collected from January 1, 1997 through May 27, 1998. Figure 5 shows the number of WWW page hits, program requests, and MBone group joins for the IMJ. During one particular day when the IMJ was first announced there were more than 2500 hits. However, because the IMJ forces page re-loads every 5 minutes (to update the schedule), this number represents an upper limit to the number of different people who visited the WWW page. The graph showing MBone group joins represents a total number of joins per day for four multicast groups: Channel 1 audio, Channel 1 video, Channel 2 audio, and Channel 2 video. A final interesting point is that both the program requests and group joins graphs show what appears to be similarly sized and similarly occurring spikes in activity. Experience in MBone use[24] suggest these spikes are due to higher use during the week and lower use during weekends. The MBone tends to reach viewers only at work and activity during the weekends tends to be lower.

While Figure 5 shows overall usage of the MBone it does not give insight to the number of different people accessing the IMJ. In Table 2 we count only the number of unique IP addresses for hosts that access the IMJ page, make program requests, and/or join one of the IMJ multicast

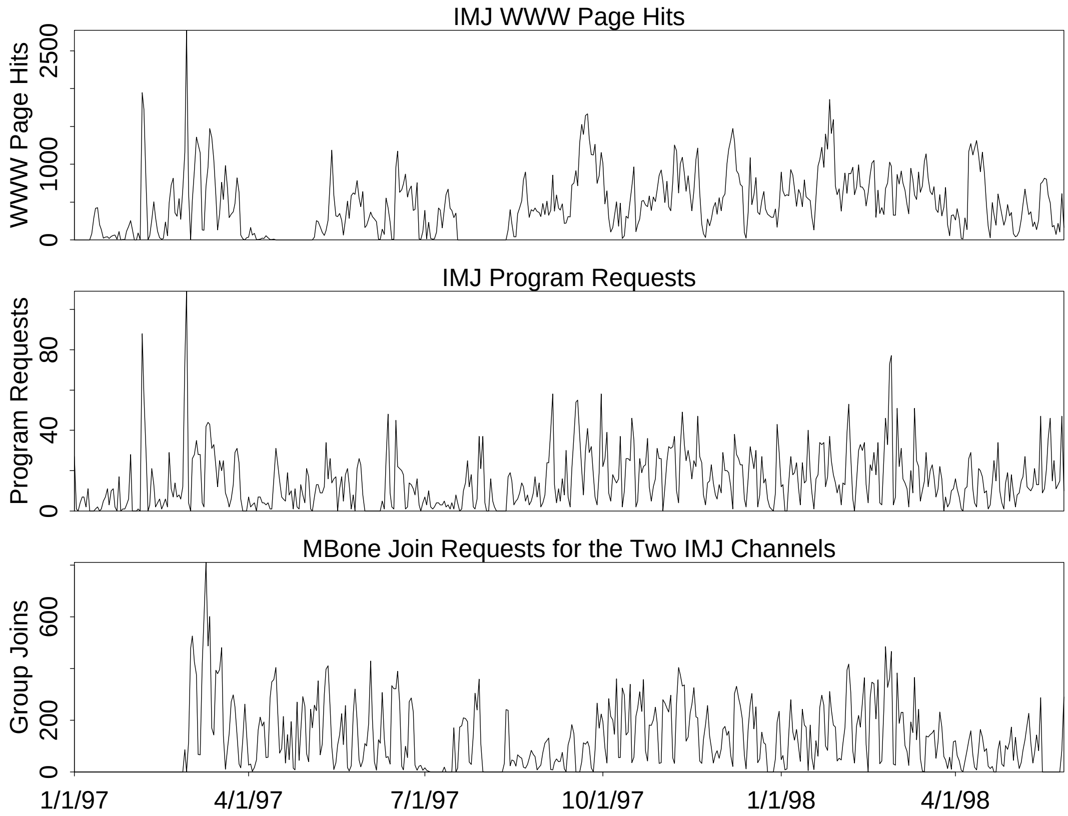

Figure 5: Daily counts for WWW page hits, program requests, and channel viewers.

groups. Results are presented for the total IP addresses, average number of accesses and maximum number of accesses for each category. Table 2 also shows a quantile breakdown for access to the MBone. This gives us a feel for how many people use the IMJ on a consistent basis. We expect the top 1% of people accessing the IMJ can be considered as frequent users. Table 2 suggests that our frequent users have hit the WWW page 807 or more times over a 512 day period (1-Jan-97 to 27-May-98), have made at least 27 program requests, and have joined one of the IMJ multicast groups at least 53 times.

Monitoring Category	Unique IP Addresses	Average Per Host	Max Per Host	Quantiles 99%	95%	90%	85%	80%
WWW Page Hits	5796	40.5	6865	807	127	39	21	13
Program Requests	2212	3.5	330	27	10	7	5	4
MBone Group Joins	6472	6.8	2502	53	22	14	10	8

Table 2: Unique hosts using the IMJ.

Table 2 suggests there is an obvious skew in the access patterns by IMJ users. Figure 6 plots these distributions, and sample Zipf Distributions with skew factors that most closely fit each distribution[25]. The four graphs in Figure 6 (from left to right and top to bottom) show the access distribution for IMJ WWW page hits per host, program requests per host, IMJ program requests per program, and group joins per host. The y-axis is plotted on a log scale and the x-axis represents the number of unique items graphed. For all but the IMJ program requests distribution, some variation of a sample Zipf Distribution fits reasonably well. We believe that the program requests would also follow a Zipf distribution if shorter snapshots of usage had been taken. Because the data in Figure 6 covers a year and a half it lumps together variations due to changes in the IMJ library including both additions and deletions. Furthermore, some of the IMJ library (IETF meetings) becomes out-dated and less popular as time passes.

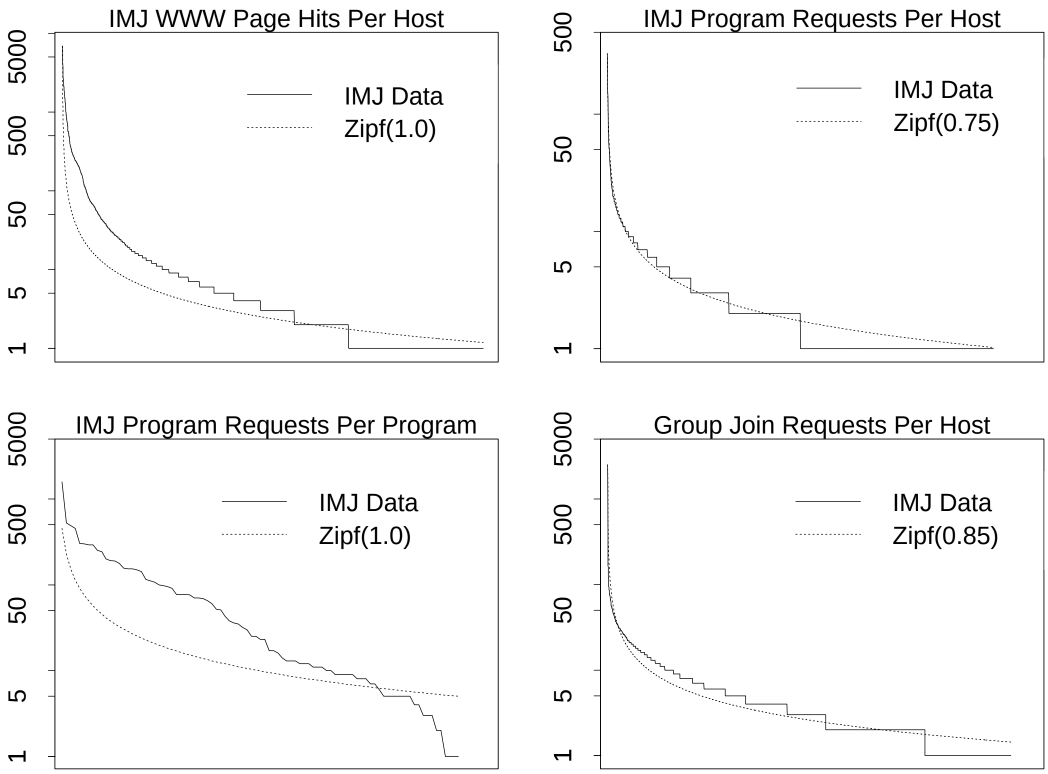

Figure 6: Plot of IMJ access distributions (including fits to Zipf distribution).

4 Simulated Performance of Large-Scale Jukebox Systems

In order to study the performance of larger scale jukebox systems, it was necessary to create a simulation environment. Performance analysis was conducted using average wait time as the key performance measure. In addition to examining the performance of large-scale jukebox systems, a secondary goal was to study the usefulness of alternate scheduling policies including voting and alternate user choice. The simulation system is based on customer behavior information derived from IMJ usage statistics. Table 3 lists the simulation parameters, range of values tested, and a set of "nominal values" which might be characteristic of a typical large-scale jukebox system.

Parameter	Range of Tested Values	Nominal Value
Simulation Events	Nominal value only	500,000
Request Arrival Rate (requests per hour)	25 to 3600	150
Number of IMJ Channels	10 to 250	100
Number of Programs	50 to 5000	500
Program Length	Nominal value only	Uniform between 5 min and 2 hours
Movie Selection Distribution	Zipf Distribution Skew Value 0.75 to 1.0	Zipf Distribution Skew Value = 1.0
Minimum Votes Required to Schedule	1 to 10	1
Number of User Alternatives to Check	1 to 20	1

Table 3: Simulation parameters and range of values tested for jukebox simulations.

Figure 7 shows a histogram of the wait times for a jukebox simulation using the nominal set of system parameters. The nominal system uses the nominal parameter values from Table 3 and implements the shortest wait time scheduling policy. The average wait time is a reasonable 11.7 minutes, and the maximum wait time experienced by a user is 34.25 minutes. The histogram shows that the results are well behaved around the average wait time.

Figure 8 is a three dimensional graph showing the tradeoff between the number of channels and the request arrival rate. The graph has two key results. As additional channel capacity is added to the system, the average wait time decreases significantly, but begins to flatten as the wait time approaches 0. The second observation is that as the request rate increases so does the average

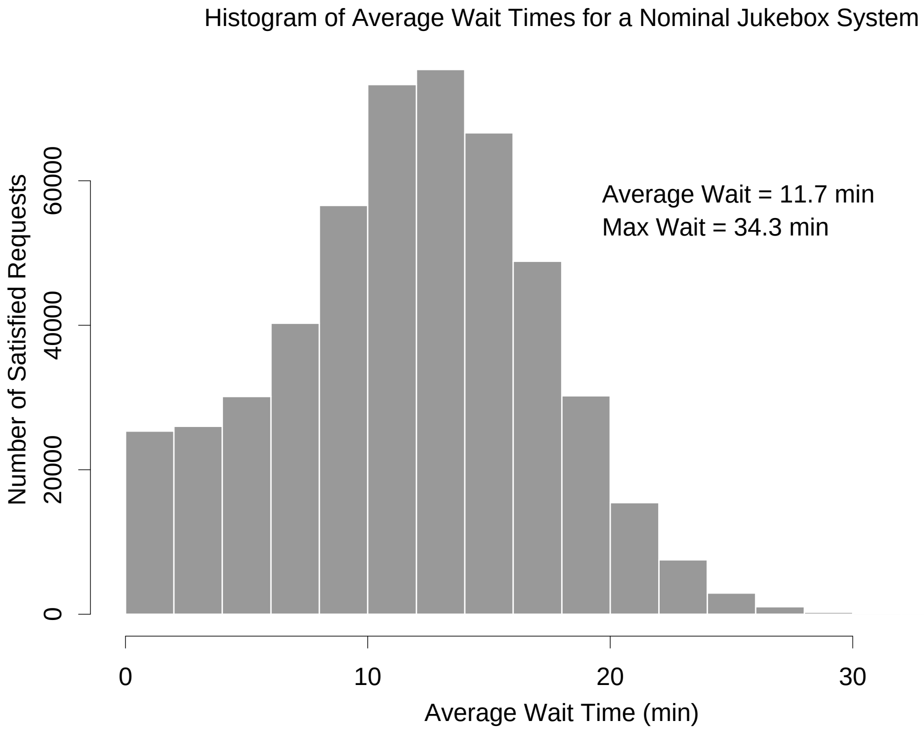

Figure 7: Histogram of wait time results for the nominal simulation case.

wait time. More importantly, the average wait time also tends to flatten out. Experience with the jukebox system suggests that the point at which the average wait time plateaus depends on the number of programs in the jukebox library. For systems with large numbers of requests per hour, the likelihood that new requests will be made for a program already scheduled is much higher for small libraries. The impact of library size is further considered in the next graph.

Figure 9 shows how the average wait time increases as the number of programs in the jukebox library increases for three different skew values of the Zipf Distribution[25]. A skew value of 0 is equivalent to a uniform distribution and a skew value of 1.0 has a highly skewed distribution (see Figure 6 for sample Zipf Distributions). Our experience suggests this tends to be the key parameter affecting when the average wait time plateaus for large scale systems. The more programs in the library and the lower the skew value, the less likely that multiple requests will arrive for the same program. For a system with only a few programs and a very high skew, the scalability can be very favorable. This characteristic is supported in on-demand style systems like the jukebox

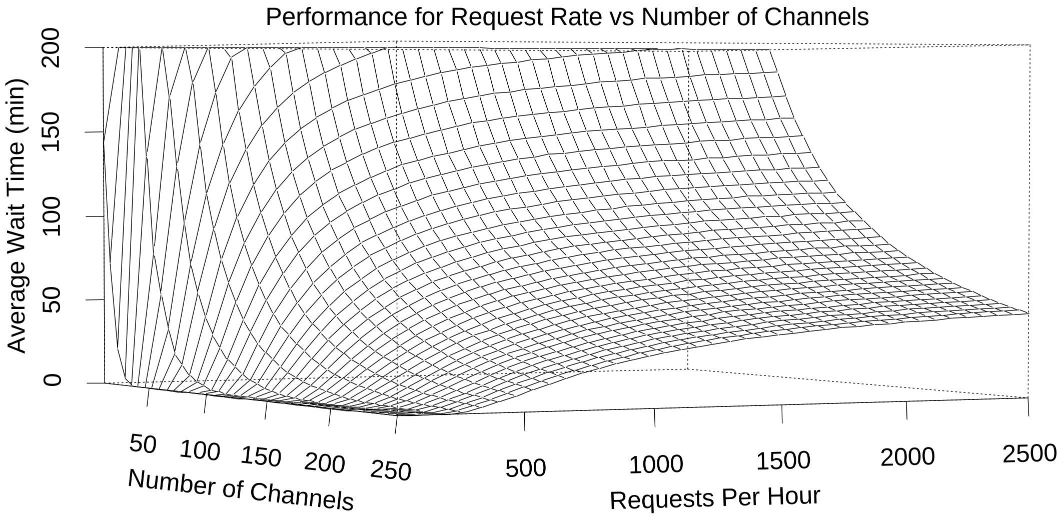

Figure 8: System performance for systems with higher request rates and channel capabilities.

because statistics suggest request patterns adhere closely to a Zipf Distribution with maximum skew. Typically, a small percentage of programs are hot and receive a majority of requests, while the remaining programs are cold and receive only a few requests.

We now turn our attention to the impact of instituting a voting procedure for IMJ pages. While previous results in this section have shown that the average waiting time for a nominal system is well behaved, users in more heavily loaded systems may experience more lengthy wait times. As the number of requests grows, users may be dissatisfied with the amount of time they have to wait for their particular request. As a result, a service provider may be interested in only scheduling requests that will satisfy a significant group of viewers. Since profit is a major motivating factor, a service provider may choose to quickly satisfy a majority of customers at the cost of inconveniencing a few. In other words, it might be to the benefit of a service provider to more quickly service requests for hot programs for which there are likely to be many interested viewers than to service requests for cold programs. The bottom line is that scheduling is a tricky and inexact business. Regardless, voting may have its uses and can be implemented in the jukebox system.

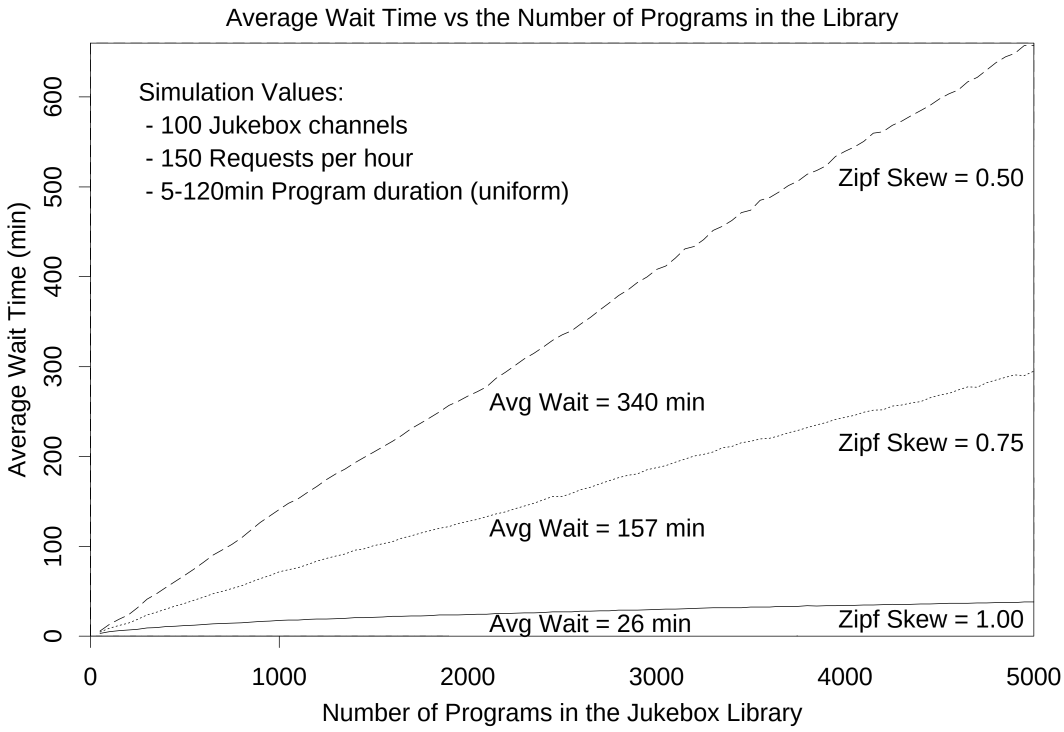

Figure 9: Effect of library size in a jukebox system with nominal system parameters.

Voting is implemented in a jukebox system by requiring some minimum number of votes to be received before scheduling a request. If however, there are free channels when a request is received, no voting threshold is used. This condition eliminates the scenario where requests are waiting to be serviced and idle channels exist, but the channels are forced to remain idle because insufficient votes have been made for a program. The system will always schedule waiting requests if there are idle channels. Voting is only used when a system is at capacity and needs to make scheduling choices.

One assumption we make in our simulations is that users are well behaved. In the real world users might simply make multiple requests for their program in the hopes of registering enough votes to cause their request to be scheduled. Our assumption might be implemented by limiting users to one vote per program and then authenticating votes. Another alternative is to enforce reasonable behavior through pricing. Most users will be unwilling to make and pay for multiple requests. But the free market system suggests some users may be willing to register multiple votes in order to expedite the scheduling of their request. In this case, making N requests (where N is the vote threshold) would cost N times the cost of a single request.

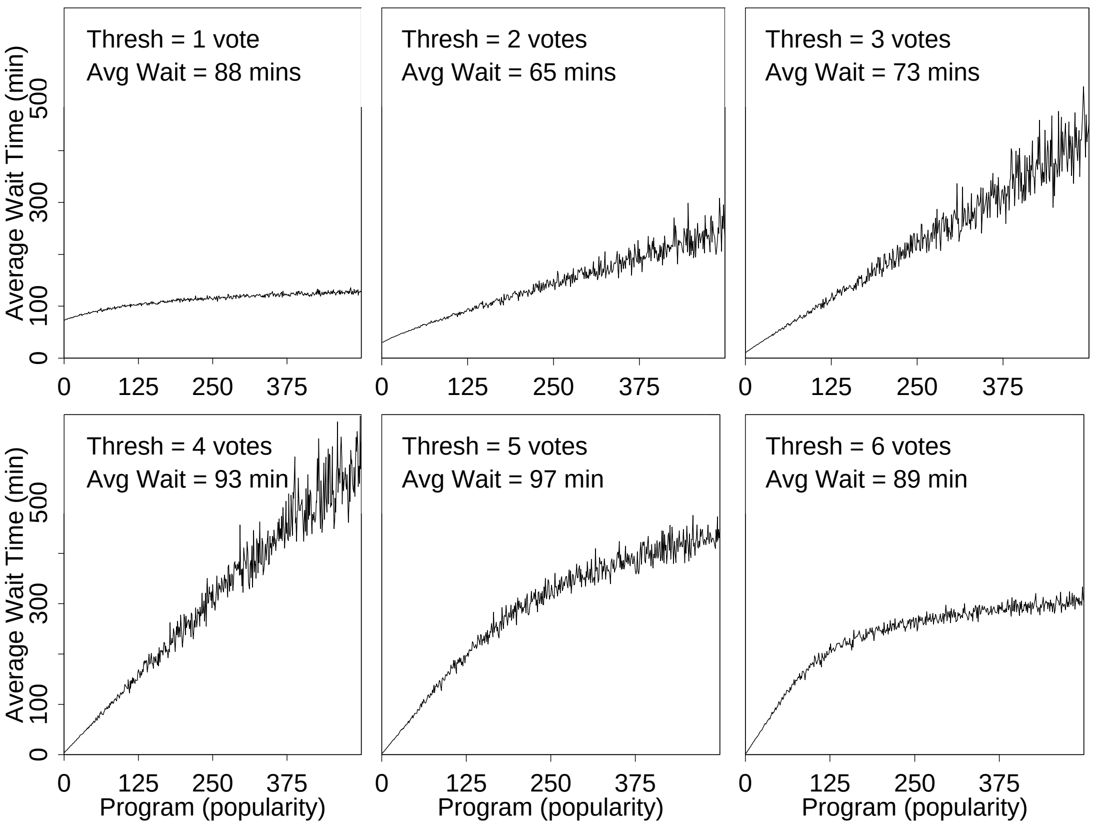

Figure 10: Effect of voting requirements in IMJ scheduling.

Figure 10 shows the results of requiring a minimum number of votes to schedule a program. The nominal set of system parameters were used, except that the request arrival rate was increased to 500 requests per hour. This increases the average wait time to about 90 minutes. As the first graph in Figure 10 shows, the wait time for each of the 500 programs in the jukebox library is roughly the same. However, as the subsequent graphs show, by increasing the voting threshold, the average wait time initially goes down; the average wait time for hot programs decreases considerably, but the tradeoff is higher wait times for cold programs. This trend continues until it peaks for a system which requires 4 votes to schedule a program. After that, the trend starts to flatten out. At this point, not enough requests are made to make the vote threshold before a channel becomes available.

The final scheduling possibility considered is based on encouraging users to select an alternate program in the hopes of reducing their waiting time. For each value in the range of the "alternative choices" parameter, a simulation run is made. In each run an increasing number of alternate choices is considered. Since we are operating in a simulation world we can more easily ask "what if" questions. In this case, we can ask the question, "If a user was made to think of an ordered list of N program choices, and their criteria for picking a program was only to minimize wait time, how much could system performance be improved?" Our simulation environment has a function that allows the server to determine how long a user making a request would theoretically have to wait without actually scheduling the request. The simulator can determine the waiting time for each of N user choices and then schedule only the request with the shortest waiting time.

One caveat to the consideration of alternate user choices is our limited ability to more exactly evaluate user attitudes. Users' decision to pick one alternative over another is not always based purely on wait time. If the time saved by making a secondary choice is minimal, a user might stick with their first choice. Furthermore, one advantage of the jukebox system is the ability to influence users to watch programs that are already scheduled. If a guide to currently scheduled programs can be displayed, users who don't necessarily know what they want to watch might be influenced to watch something that is starting soon. An analogy can be made to people who go into a movie rental store without knowing what they want to rent, or those who turn on the TV without knowing what they want to watch. Browsing and channel surfing are powerful tools that can influence people's decision on what to watch.

Figure 11 shows the results for systems which consider multiple user choices. The nominal system parameters are used, but the request arrival rate is set to 500 requests per hour. As such, the average wait time with only one user choice is almost 90 minutes. Significant performance gains can be made if users are encouraged to consider alternate choices. Even if only two choices are considered, the average waiting time drops to only 25 minutes. In the case of two choices, 60% of first choice requests are satisfied and only 40% of the users making requests accept their second choice. This tradeoff seems reasonable given the significant improvement in average wait time. As the number of user choices considered is increased, the gains made in average wait time start to flatten. Considering more than four choices yields little in terms of improved average wait time.

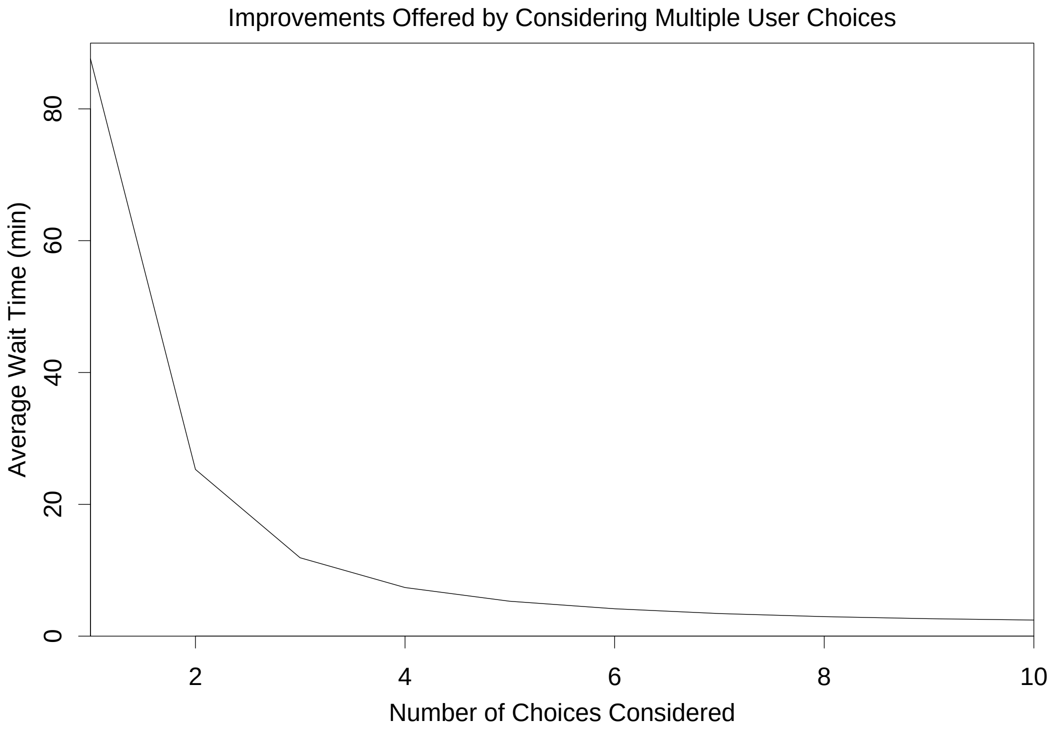

Figure 11: Effect of having users request programs using multiple choices.

5 Advanced Jukebox Services

The jukebox paradigm and the IMJ implementation have created a number of challenges in a number of areas. While some of these issues have been explored for related paradigms, the jukebox paradigm requires a re-examination of some features. In this section we concentrate on describing issues that are either already provided or that we expect to provide in the IMJ system.

5.1 Providing Interactivity

Because the jukebox is inherently a one-to-many service, it does not provide a specific mechanism for individual viewer interactivity. However, if sufficient network and server resources exist, VCR-like functions can be provided. One of two types of interactivity can be provided:

- **Limited Interactivity**: Instead of allowing a viewer continuous playout control, VCR actions are only allowed in increments. This allows interactivity to be provided using only a small number of additional groups at specific offset intervals. This type of interactivity is comparable to a similar concept proposed for near VoD systems[1]. While the advantage is that fewer resources are required, the disadvantage is that playout control is more coarse-grained. Also, this type of interactivity is only useful for large systems were interactivity is expected to occur frequently.

- **Full Interactivity**: Full interactivity can be provided using one of two methods. The first method requires the server to split the viewer from the main IMJ channel and provide the viewer with a single stream. When the first VCR action is initiated, resources are allocated to the viewer for the remainder of the program. At the end of the program, the viewer returns to the real-time jukebox schedule. The second method, and the method we are implementing in the IMJ, is to put the processing burden on the viewer. The viewer is responsible for buffering the server's transmission. Once buffered, the viewer can initiate VCR-style functions. The server transmits at a constant rate, and viewers use buffering to capture the program stream but then may be watching different parts in the program. The advantage of this method is that the processing burden is removed from the server. Furthermore, the amount of buffer viewers are willing to dedicate to a stream dictates the freedom with which they can move around within a stream.

5.2 Distributed Server and Interface

As has been discussed the IMJ uses the WWW as a request interface and program schedule display interface, and then the MBone for program playout. We are investigating improving the performance of the IMJ by making these two services distributed. The program playout aspect is already very much distributed. Content is not simply stored on a single machine, but spread across the local disks of a number of workstations. Depending on the category of content the IMJ scheduler will make a request to the particular machine that the content is stored on, and that machine becomes responsible for playout. Currently, we are using three different machines as servers, two Sun and one SGI workstation. The ability to act as a server is relatively hardware and operating system independent, and the only requirement is a compiled version of *rtpplay*[23] and sufficient local disk space.

There are a number of ways in which the request scheduling component of the server can be made to be distributed. Each of these methods can potentially lead to a shift in the jukebox paradigm. Several of the possibilities are briefly described as follows:

- **Separate pages**: This option actually creates multiple separate jukebox systems. So, in fact there is really no distribution of scheduling, just separate services.
- **Common schedule**: Multiple WWW pages are established at various sites but all requests are sent to a common scheduler which makes the scheduling decision and then redistributes an identical schedule to all WWW servers. The advantage of this system is that it reduces the centralization of WWW page accesses. However, the scheduler is still potentially a single point of failure.
- **Invisible distributed systems**: A more difficult option to implement would be an automatic system with an "anycasting" style interface. Users attempt to view a page at a global, common URL (http://imj.edu/ for example). This address is actually a server pointing the user to the "closest" local IMJ server. For example, users on the West Coast might go to the WWW page at http://imj.ucsb.edu/ while users on the East Coast go to http://imj.gatech.edu/. Each WWW site has its own scheduler and its own channels. In the ideal case, sets of channels are non-overlapping and users are only ever serviced by a single IMJ server. In practice, this type of system will likely be difficult to implement for a number of reasons, including identifying good server locations, dealing with imprecise TTL scoping, and routing in complex network topologies.

5.3 Advanced Reservations

Instead of scheduling programs as soon as possible, a service provider may allow viewers flexibility regarding when a program is scheduled to start. Users may be allowed to specify that a program start at some time beyond the current schedule. Advanced reservations are a nice feature but they create two problems. The first problem is an issue of fairness. Allowing scheduling arbitrarily far in advance creates the potential for viewers to abuse the system and schedule programs without really knowing whether they will be around to watch them. The second and more interesting technical problem is how to deal with "dead periods". Dead periods occur as the playout time of a program approaches. Consider for example, a program scheduled in advance to start at 8:00pm. As the on-demand schedule fills and approaches 8:00pm, there will come a time when a requested program will be too long to fit in the period between the last on-demand program and the 8:00pm reservation. The schedule is left with a gap in which no program will fit. The solution to this problem involves using three techniques:

- **Schedule programs with an awareness for the mean and median program length**: In some cases, a program may be scheduled on an alternate channel if it creates a high probability of a significant, unfillable dead period.
- **Automatically schedule a program that fits into the available time slot**: If there is a program in the library which would nicely fill an existing dead period, go ahead and schedule it even though no viewer has requested it. Any program is better than no program.
- **Adjust the start time of the advanced reservation**: If a dead period is only a couple of minutes or seconds shorter than a requested program, the reservation could be pushed forward slightly. Or, if the dead period is too long, the reservations could be moved up. However, violating promised start times may cause dissatisfaction among some viewers, but slight scheduling changes may not be too bad.

5.4 Service Pricing

The cost of using the jukebox system is an important consideration for a service provider. From a technical perspective, pricing can be used as an equalizer. It can be used to control viewer behavior, discouraging actions which require precious system resources, and encouraging behavior which allows more viewers to be satisfied. A typical pricing scheme might include two types of charges:

- **Monthly access charge**: Simple access to all the jukebox channels is worth something. The more channels and the better the content the more a service provider can charge. A service provider offering a number of broadcast and on-demand channels could easily charge current cable TV rates. Monthly access charges will likely account for a majority of a system's revenue.

- **Per-request charge**: There can be a per-request charge each time a viewer makes a request. The dilemma here is making this charge small enough to encourage viewers to make requests but large enough to discourage viewers from randomly requesting programs that they have no real intention of watching. Without market testing we cannot attempt to guess what viewers would be willing to pay.

There are also numerous opportunities for service providers to charge additional fees for premium services. Additional services include advanced reservations, server-based interactivity, access to premium content like new release movies, premium quality including a high definition (HDTV) version of a program, etc.

6 Concluding Remarks

This paper discusses our proposed jukebox paradigm, an exciting alternative to other video-on-demand paradigms. The jukebox paradigm we propose is based on the premise of allowing any viewer to watch any other viewer's requested program. Program requests are scheduled on one of a system's set of channels using a set of scheduling policies. Any viewer who wants to watch a program on a particular channel simply "tunes" to that channel. Content on each channel is delivered from a server to all viewers watching that particular channel. This paper also describes our efforts to prototype a jukebox system called the *IMJ*. The IMJ provides scheduling via the WWW and content delivery via the MBone. Through the IMJ we have collected a number of sets of usage data which we have analyzed to give us feedback on the success of the jukebox paradigm. All indications suggest that the jukebox is an easy-to-understand model and has been accepted into the MBone community as a useful service. Taking the jukebox paradigm a step further we have built a simulation environment to evaluate the performance of larger scale systems than the IMJ. The overall conclusion is that the jukebox offers a combination of scalability and flexible scheduling not found in other paradigms. As with any system that gains scalability through request aggregation, the size and access distribution of the program library has a big impact on performance. Again, the jukebox paradigm offers an advantage by giving service providers the opportunity to influence user choices and further improve performance.

References

[1] K. Almeroth and M. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1110–1122, August 1996.

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

[3] S. Casner, *Frequently Asked Questions(FAQ) on the Multicast Backbone(MBone).* USC/ISI, December 1994. Available from ftp://ftp.isi.edu/mbone/faq.txt.

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

[5] H. Schulzrinne, S. Casner, R. Frederick, and J. V., "RTP: A transport protocol for real-time applications," Tech. Rep. RFC 1889, Internet Engineering Task Force, January 1996.

[6] W. Holfelder, "Interactive remote recording and playback of multicast videoconferences," in *4th. International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS '97)*, (Darmstadt, GERMANY), September 1997.

[7] A. Klemets, *The Design and Implementation of a Media on Demand System for WWW.* Geneva, SWITZERLAND, May 1994.

[8] R. Vetter and C. Jonalagada, "Multimedia system for asynchronous collaboration using the multicast backbone and the world wide web," in *Proceedings of the Annual Conference on Emerging Technologies and Applications in Communications*, (Portland, OR, USA), pp. 60–63, May 1996.

[9] P. Parnes, M. Mattsson, K. Synnes, and D. Schefstrom, *mMOD: the Multicast Media-on-Demand System.* Centre for Distance-spanning Technology, May 1997. (submitted).

[10] X. Li and M. Ammar, "Bandwidth control for replicated-stream multicast video distribution," in *High Performance Distributed Computing (HPDC)*, (Syracuse, New York, USA), August 1996.

[11] S. Cheung, M. Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution," in *IEEE Infocom*, (San Francisco, California, USA), March 1996.

[12] J.-C. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *ACM Sigcomm*, pp. 58–67, September 1994.

[13] X. Li, S. Paul, P. Pancha, and M. Ammar, "Layered video multicast with retransmission (LVMR): Evaluation of error recovery schemes," in *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video '97*, (St. Louis, MO, USA), May 1997.

[14] K. Almeroth and M. Ammar, "Scalable delivery of web pages using cyclic best-effort (UDP) multicast," in *IEEE Infocom*, (San Francisco, California, USA), June 1998. (to appear).

[15] T. Liao, "WebCanal: a multicast web application," in *Sixth International World Wide Web Conference*, (Santa Clara, California, USA), April 1997.

[16] P. Parnes, M. Mattsson, K. Synnes, and D. Schefstrom, "The mWeb presentation framework," in *Sixth International World Wide Web Conference*, (Santa Clara, California, USA), April 1997.

[17] R. Clark and M. Ammar, "Providing scalable web service using multicast delivery," in *IEEE Workshop on Services in Distributed and Networked Environments*, (Whistler, CANADA), June 1995. (to appear in *Computer Networks and ISDN Systems*).

[18] R. El-Marakby and D. Hutchinson, "Integrating RTP into the world wide web," in *World Wide Web Consortium Workshop on Real Time Multimedia and the Web*, (Sophia Antipolis, FRANCE), October 1996.

[19] M. Handley, "Applying real-time multimedia conferencing techniques to the web," in *World Wide Web Consortium Workshop on Real Time Multimedia and the Web*, (Sophia Antipolis, FRANCE), October 1996.

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

[21] 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, January/February 1992.

[22] A. Chervenak, D. Patterson, and R. Katz, "Storage systems for movies-on-demand video servers," in *IEEE Symposium on Mass Storage Systems*, (Monterey, California, USA), pp. 246–256, September 1994.

[23] H. Schulzrinne, *README for RTPtools*. Columbia University, August 1997. Available from ftp://ftp.cs.columbia.edu/pub/schulzrinne/rtptools/.

[24] K. Almeroth and M. Ammar, "Multicast group behavior in the Internet's multicast backbone (MBone)," *IEEE Communications*, vol. 35, pp. 224–229, June 1997.

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