From Broadcast Television to Internet Audio/Video: Techniques and Tools for VCR-Style Interactivity

David B. Makofske
Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
davidm@cs.ucsb.com

Kevin C. Almeroth
Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
almeroth@cs.ucsb.edu

October 1999

Abstract

One of the new applications evolving in the Internet is streaming audio/video. A major reason for its growing popularity is interest in the compelling new services that become possible. Prototype services are being developed which are new to the Internet but offer the same look, feel, and functionality that have traditionally only been found in services delivered via other communication medium, e.g. broadcast television. In addition, the Internet is evolving to offer "value-added" services, like streaming audio/video with VCR-style interactivity and embedded hyperlinks. We are poised both on seeing the development of new paradigms for interacting with audio/video, and on seeing the merging of broadcast television and Internet-based broadcasts. Before this process can be considered successful, a number of technical challenges, derived from the various ways in which content is physically delivered, must be solved. In this paper, we focus on the value-added service of VCR interactivity. VCR interactivity has long been a challenge for both broadcast television and streamed Internet audio/video. The challenge is how to provide individualized playout for content being streamed to a large group of users using one-to-many delivery. While some new companies are starting to offer devices which provide this kind of service for broadcast television, there are still numerous technical challenges for the Internet-based version of a similar service. This paper has a three-fold objective. First, we describe the types of services available in the traditional broadcast infrastructure and compare these to the types of services that are deployed or possible in Internet-based services. Second, we describe our attempts to implement some of the more challenging and novel service types. In particular, we examine client-based control of programs streamed over the Internet to tens, thousands or even millions of users. Finally, we discuss the impact of these services on the protocols and applications used to support Internet-based, multi-party conferencing.

1 Introduction

The term *convergence* can readily be applied to the merging of technologies for personal computers, telecommunication, and television into a single user experience. One excellent example of this phenomenon is the dramatic increase of streaming audio/video in the World Wide Web (WWW) over the past few years. Although the exact amount of available content is hard to quantify, the two most common shareware tools for viewing this content have been downloaded millions of times each. respectively. Streaming audio/video has grown into an important data type for the WWW and the Internet. One of the reasons for the growing popularity of WWW-based broadcasting is the ability to offer traditional broadcast television services combined with compelling new services.

As the use of Internet-based audio/video grows, it will have to mature in order to be able to compete with broadcast television. One of the first steps in the maturation process will need to be improved quality. High quality video and stereo audio are critical in creating production-caliber services that users are willing to pay for. A second step will be to develop services that are highly desired by large groups of users. Content providers and multimedia tool developers are in the position of having to create programs and interfaces that appeal to a wide range of audiences. In this context, the Internet is expected to become a medium similar in size and importance (i.e. revenue generation capability) to broadcast television. As a result, many of the first-generation of services being offered via the Internet are designed to appeal to traditional broadcast television watchers—these new services have a similar look, feel, and function. As the Internet-based services become popular, and the true power of the the Internet is realized, the *next*-generation of services will offer services far beyond what the broadcast television infrastructure can offer. Examples of basic first-generation services exist, as do the precursors to services with next-generation functionality. Our focus in this paper is to examine the challenges of developing and deploying a service with VCR interactivity.

Even as next-generation Internet services evolve, the goal of matching and then surpassing the same functionality found in broadcast television is not easy. Obviously the Internet has a long way to go before matching the quality and service of broadcast television, but matching these levels is not easy since broadcast television evolution has recently hit a new plateau. High Definition Television (HDTV) is becoming a reality and several companies have developed their own next-generation services. These services, one of which is VCR-style interactivity for live programs, are competing well against similar Internet services. One goal of this paper is to categorize the types of services available for broadcast television and compare these to the types of services that could be available in the Internet.

As we have already alluded, one of the first next-generation services is likely to be VCR-style interactivity for live programs. This service is currently under development for both broadcast television and Internet-based services. Using local buffering, a user is able to pause, rewind or fast forward without having to affect the single stream being sent to all users. This type of service has long been associated with Internet-based video-on-demand (VoD)[1], but recently, is being considered as a possible service for broadcast television. In a digital world it is straightforward to continually record a live stream and then allow the user to randomly access any point of the buffered stream. However, many of the limitations, until recently, have not been technical. The true problem has been the cost of storing even a small duration of the bulky audio/video stream. In the broadcast television world, there is an added challenge: the need to first convert the analog signal into a digital signal and then store the digital signal. As each of the technical issues have been solved, the possibility of offering VCR-style interactivity for streamed programs, whether delivered via the Internet or other infrastructures, has nearly become a reality.

This paper has a two-fold objective. First, to describe the types of services available in the traditional broadcast infrastructure and compare these to the types of services that are deployed or are possible in Internet-based services. Second, to describe our attempts to implement a tool that provides VCR-style interactivity for streaming audio/video being sent to tens, thousands or even millions of users. The provision of these types of services is theoretically straightforward, but the current Internet infrastructure and the utilities, protocols, and delivery methods being used have created some interesting challenges. Through the description of the tools we have developed, we explore these challenges in depth.

This paper is organized as follows. Section 2 describes program delivery paradigms used in both the Internet and in the traditional television broadcast network. Section 3 discusses mechanisms for providing user-controlled, VCR-style interactivity for Internet programs. Section 4 details our work to design and implement a client-side buffering tool, called *MControl*. Section 5 presents the implications of using such a tool for interactive, multi-party sessions. Section 6 offers related and future work. The paper is concluded in Section 7.

2 Delivery of Audio/Video Programs

In an attempt to understand how audio/video programs are delivered to customers, we compare the traditional set of services first offered via broadcast television and progress to the types of services evolving in the Internet. Included in this discussion is the kinds of services that are being offered in cable television and satellite distribution networks. In making comparisons, it is helpful to use a set of characteristics to differentiate between types of service. These characteristics are described next.

2.1 Criteria to Characterize Content Delivery Systems

There are a large number and wide variety of applications offering delivery of audio/video programs. These applications are themselves used in a wide variety of services, and these services vary greatly depending on the infrastructure used to deliver the audio/video. Fundamentally different paradigms are used for broadcast television networks than are used in the Internet. Even the terminology is different. In an attempt to provide a common basis for comparison, we first identify the common objective of these services. The objective is to deliver video programs with accompanying audio signals from a transmission site to one or more users simultaneously. Programs may be delivered via a variety of media including terrestrial air waves, cable infrastructures, satellite-based networks, or the Internet.

The key distinction among the various content delivery services is simply the way in which the programs are delivered. In the past, other distinguishing characteristics could be used, like encoding format, quality level, analog/digital transmission, etc. For example, in the United State, broadcast television over terrestrial airways was always delivered in the National Television Systems Committee (NTSC) format. More recently, new formats like HDTV are being used. Another example is satellite-based networks which can now encode audio/video using the Motion Picture Experts Group (MPEG-2) standard and transmit it using the Digital Video Broadcast (DVB) system[2]. And while the Internet has always used the Internet Protocol (IP) as the basic delivery mechanism, the encoding formats have varied widely from MPEG-1 and MPEG-2 to lower bit-rate encoding schemes like H.261. Therefore, the only true distinguishing characteristics between content delivery systems is the infrastructure itself. Before considering the set of criteria used to differentiate these delivery systems, we first identify the types of content delivery infrastructures that we consider in this paper:

- *Terrestrial Broadcast Television*
- *Cable-Based Broadcast*
- *Satellite-Based Broadcast*
- *Internet-Based Transmission*
- *Physical Media Delivery*

The set of delivery infrastructures is relatively straightforward except possibly for the last. We include for consideration in this last type the use of video cassettes and now digital video disks (DVDs). While not exactly a "delivery infrastructure", tapes and DVDs that have been purchased or rented represent a common way of watching audio/video programming. Furthermore, given that we focus on VCR-style functionality, it only makes sense to include tapes and DVDs in our discussion.

The most basic function of the various delivery infrastructures is signal delivery. However, there are other factors that affect the kind of service offered to users. In many cases, the history of a particular infrastructure has dictated the kind of service offered. For example, broadcast television has always been a small number of channels with programming scheduled far in advance. Television watchers have become used to the long established way in which television programs are selected for broadcast. But as the number of ways in which programs can be delivered grows, and as the desire to give users better, different, and new service grows, the fundamental characteristics of the service will change. We now identify the set of characteristics that can be used to differentiate the various kinds of service being offered over the various types of delivery infrastructures. The set of characteristics we have created are:

- **Content Determination** – A decision must be made on what content to play. The decision is made by either the channel owner, the infrastructure owner, or the user. A user decision usually means programming is on-demand and scheduling is dynamic. Otherwise, the schedule is likely to have been determined far in advance and is relatively rigid.
- **Content Liveness** – Content can originate either from a live event or from a storage device.
- **Network Paradigm** – Content can be delivered using different delivery mechanisms depending on what the network supports and how many users there are. One-to-one delivery (unicast), one-to-many (multicast), or one-to-all (broadcast) can be used.
- **Delivery Method** – Content can either be delivered in its entirety before playout begins, known as *download-and-play*, or *streamed* to the user on an as-needed basis.
- **Interactivity Availability** – VCR-style functions may or may not be available. Partial functionality may also be available depending on a variety of factors including how content is delivered and whether content is delivered via unicast, multicast or broadcast[3].
- **Interactivity Provisioning** – If interactivity is provided, it can be implemented by manipulating the stream at the server or controlling the stream at the client.

Attempting to describe each delivery infrastructure as a single instantiation of these criteria is impossible. The difficulty arises because some delivery infrastructures have evolved over time, and while they once did not offer a particular kind of service, it may be offered now. In the next section we focus instead on characterizing the historical evolution of the various delivery infrastructures.

2.2 Characterizing Delivery Infrastructures

Terrestrial, cable, satellite, Internet-based, and physical media delivery systems have all evolved substantially over when they were first invented. In this section we briefly discuss the evolution of each. Figure 1 shows a summary of the systems we consider and how important new functionality has evolved over time. The x-axis shows the list of major infrastructures used to deliver audio/video programming. The y-axis is a relative time axis that shows the evolution of "value-added" services on top of basic audio/video delivery.

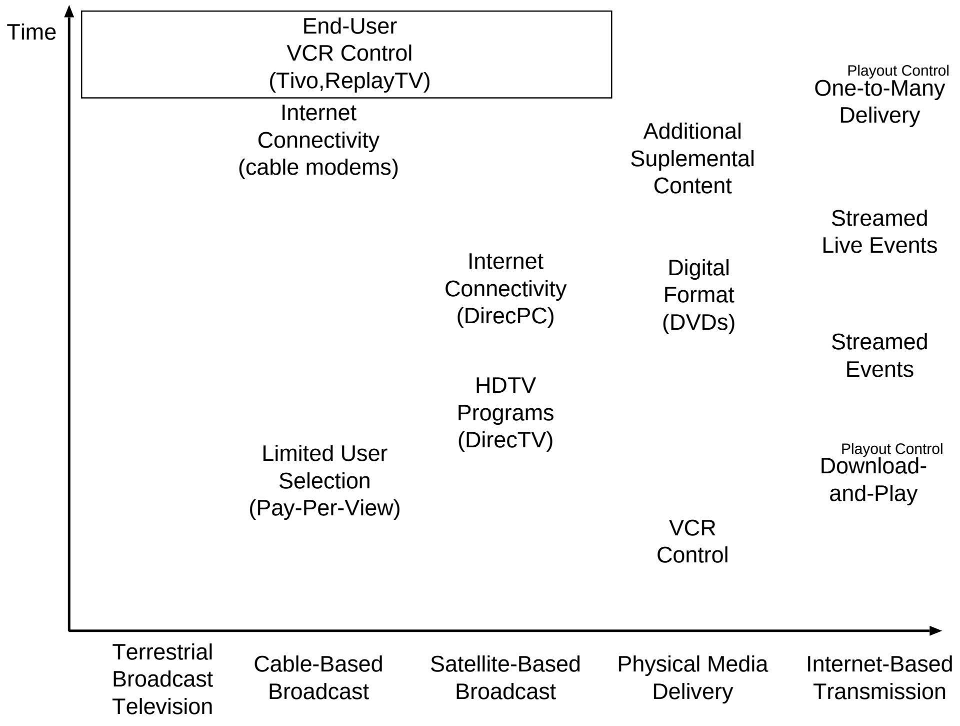


Figure 1: Network delivery infrastructures and their evolution.

2.2.1 Terrestrial, Cable, and Satellite Television

Content delivery via terrestrial, cable, and satellite networks began slowly, but more recently has started to show some interesting developments. Terrestrial television came first and decades have been spent improving the quality of the signal. In the first days, television was a single channel of black and white video and no audio. Recently, with the advent of cable and satellite systems, television quality has taken the next step. It has evolved to the point where users can receive hundreds of channels delivered digitally, in full color, and with stereo or surround sound. However, the basic broadcast paradigm has not changed. Only very recently have content providers operating in each of three delivery infrastructures started to try to differentiate their service. As competition has increased, content providers have tried harder to find ways to lure customers.

The original model for choosing which content to deliver on a channel was made by those who owned the rights to transmit on the channel. Recent attempts to improve service have been focused on creating the illusion that customers have the ability to select content on-demand and then have it delivered exclusively to them. But, these types of services are simply built on top of the same basic broadcast model that was used when television was first introduced. The most widely available service to-date that provides this kind of service is Pay-Per-View (PPV)—typically only available in cable-based systems. While the user does not have a choice in what content is played, there is the illusion that customers can pick from one of several movie times. The ultimate next-step would be to provide customers true video-on-demand. Today, this kind of service is only available in hotels, and even then, only to a limited degree. Another option that is becoming popular offers even less control than PPV. Because of the time differences between the East Coast and the West Coast of the United States, some cable and satellite operators are carrying what normally is only the East Coast feed or only the West Coast feed to the entire US. The service that this provides is to give customers a 3 hour time shift of the same program. Programs that are normally on at 8:00pm on the East Coast are available at 8:00pm and 11:00pm, and on the West Coast the same program is available at 5:00pm and 8:00pm. The philosophy here seems to be more control through more choices through more channels.

One of the new services that has only been deployed as a test service and on a very limited basis, is true interactive TV. This service allows users to get additional information about what they see on the television screen by simultaneously accessing the Internet. This service exists in a number of service provider trial markets and spans the range of terrestrial, cable, and satellite providers. Some of the earliest trials occurred in the early 1990s; the most famous of which is probably Time Warner's attempt to provide true video-on-demand to customers in Orlando, Florida. However, for reasons of complexity, cost, and a lack of profitability, these services never got of the ground commercially.

The latest service, being offered by two companies: TiVo (http://www.tivo.com/) and ReplayTV (http://www.replaytv.com/), is to use a set-top box to provide localized control of a television stream[4]. In addition to the ability to provide VCR-style control of a live stream, both services also allow a user to record and store up to some number of hours of video. Current systems store around 30 hours of audio/video digitized and compressed into the MPEG-2 format. The basic operation of these systems is to take a received audio/video stream; convert it to MPEG-2; and either play the live stream or some part of the buffer to the television set. The set-top box has the flexibility to handle audio/video from any media whether it is from a terrestrial, satellite, or cable system. Some of the companies offering this service also offer additional features like the ability to randomly record programs considered to be of interest to the user.

2.2.2 Physical Media Delivery

The use of physical media has played an interesting role in affecting the evolution of services in other infrastructures. After becoming popular, matching the flexibility and services provided by video tapes and DVDs became the goal of broadcast television. Video-on-demand was created to emulate the user-controlled selection of content. The ability to pause, rewind, or fast forward became the ultimate value-added service for streamed content. Only in the Internet where programs could be fully downloaded and accessed locally was this kind of functionality available. With the advent of PPV and services like TiVo or ReplayTV, the services of physical media have begun to be challenged.

A more recent development in the physical media delivery arena is the use of DVDs as an alternative to video cassettes. DVDs allow users to watch HDTV programs with the highest available audio and video quality. Again, content providers using other infrastructures are struggling to keep up. DVDs also provide additional film footage, movie facts, information about the actors/actresses, etc. Some DVDs, because they can be run from PCs or laptops, can use Internet connections to give the user even more dynamic content. Like VCR-style interactivity and quality, these new services have driven the demand to create new opportunities for interaction through broadcast television.

2.2.3 Internet-Based Transmission

Evolution of programming in the Internet has evolved in two orthogonal areas. The first efforts were simple download-and-play application and the content was mostly MPEG-1[5] video[6, 7]. Originally, files were downloaded using the File Transfer Protocol (FTP) and played locally, without sound, and without VCR-style interactivity. Like television, these simple beginnings have evolved over time into sophisticated services. However, unlike television's relatively static delivery paradigm, the Internet enables a wide variety of possible services. Via the Internet, users can receive low or high quality audio and/or video; can receive files or streamed content; can receive live events; and can have a range of VCR-style interactivity. Not all combinations of all services are yet available but deployment is a matter of a willing service provider and compelling content. Awareness and demand for such applications have steadily increased.

Effort in the second area of Internet programming has been focused on providing content and services using multicast communication[8]. Initial efforts provided tools and network support to deliver meetings and live events to hundreds and even thousands of clients simultaneously. Even though the initial focus was on conferencing, other applications like video-on-demand[9] and distance learning[10] are emerging. Because multicast is specifically targeted at scalable delivery to a nearly unlimited number of clients, the challenges are now to incorporate value-added services like interactivity[3] and on-demand scheduling[11, 12].

3 Providing Interactivity for Internet Programming

Providing interactivity for Internet programming has been the focus of numerous research efforts in the past few years. Table 1 summarizes a few of the more prominent systems for which interactivity has been studied.

Early research by Gelman[13] looked at interactivity for stored data sent by unicast. Thanks in part to this work, this is no longer a particularly difficult research problem and several commercial tools provide this capability. Such functionality can be implemented using one of two techniques. One solution is to make the client responsible for buffering frames and responding to user commands. The other solution is to use a control channel between the client and server and have the server

Category	Stored		Live	
	Unicast	Multicast	Unicast	Multicast
Interactivity Availability	Full	Limited or Full	Not Available	Not Available
Interactivity Implementation	Local or Server-based	Mostly local w/ some server	Local or Server-based	Local

Table 1: Types of interactivity provided for various types of Internet content.

alter the stream according to the user's wishes. While requiring the server to adjust the playout point is cumbersome, it is the only option if the client decoder has limited capability.

More recent research looks at the problem of providing interactivity for programs delivered using multicast. The fundamental problem is that the server cannot alter playout to satisfy one user without adversely affecting other users. At the time this research was conducted, local buffering was prohibitively expensive and was not considered as an alternative. Several proposals were made that work on the principle of splitting a single group into multiple streams. When a user initiates a VCR action, a message is sent to the server via a control channel. The server then creates a separate multicast group[3]. The major disadvantage of this approach is that these additional streams degrade the efficiency and scalability of using multicast. As a result, various optimizations were proposed which attempted to aggregate users into as few groups as possible[14, 3].

The concepts of stream splitting has been proposed both in the context of *near* video-on-demand systems and for heterogeneous multicast groups[15]. However, the authors are not aware of any commercial efforts which have adopted stream splitting as a technique for providing interactivity or for dealing with heterogeneity. Even in the multicast backbone known as the MBone, there have been few tools that provide these services. As a result, even though this work has been evaluated in the research community, little effort has been given to studying the feasibility and performance of these techniques.

The final types of service included in Table 1 are based on the delivery of live programs. To a certain extent, live programs are very similar to stored programs. However, there are a few subtle differences which make the provision of interactivity a significantly different problem. In particular, two characteristics of live programs distinguish them from stored programs:

1. Live programs are happening in real-time; therefore, interactivity is limited by the laws of physics. Obviously a user cannot fast forward into the future. This fact alleviates the complexity of having a streaming server handle users' desire to fast forward beyond the server's current playout point.

2. Live programs tend to be more interactive and have stricter requirements for minimizing delay. Examples include distance learning which is based on an audience receiving a presentation

but also enabling audience members to participate. Because the stream is live and users do not want to fall behind, the idea of providing interactive functions seems less applicable. This is certainly not the case. The compelling service is for users to be able to manipulate the stream so they can replay important parts and skim (or skip) less interesting parts.

In this paper we focus on the provision of interactivity for programs streamed to a group using multicast. While the services offered by companies like TiVo and ReplayTV exist for broadcast television, we show that there are additional challenges to deploying a similar service for the Internet. We focus on providing VCR-style functions for both stored and live Internet content. In order to make this task easier we consider live and stored programs to be essentially the same. This has implications for users watching both live and stored programs. For stored programs, users are not allowed to fast forward beyond the point the server is playing; similar to how users watching a live stream cannot fast forward into the future. For live programs that are interactive, users are responsible for maintaining a sense of "keeping up" and also for not "missing anything important". Our goal is to develop a prototype of a client-side interactivity tool and then to evaluate it for MBone sessions. In particular, we look at the Interactive Multimedia Jukebox (IMJ)[12] and live broadcasts of IETF meetings.

4 The *MControl* Tool

Providing interactive functions for streaming audio/video becomes more complex when multicast communication is used. One common solution is to split a single group into multiple groups as users leave the main stream. The disadvantage of schemes using this technique is that scalability is degraded. There are then a number of improvements which can be made to reduce the number of groups needed. The general problem with this set of solutions is the added complexity. The system must have mechanisms for allowing group members to signal the server in order to change groups; the server must be able to decide when to split and re-join groups; the server must also be able to access a particular program multiple times at random playout points; and clients must be able to re-synchronize to a new stream without losing playout continuity.

A second solution is to use local buffering. While this places an additional burden and expense on the client, it eliminates complexity in numerous other parts of the system. Furthermore, local buffering can eliminate *all* other complexity if clients are (1) limited to interactivity within the stream they have buffered, and (2) are not allowed to contact the server to request stream playout changes. With a small tradeoff in functionality, interactivity can be provided with a marginal increase in complexity.

The basic idea of local buffering is shown in Figure 2. Figure 2(a) shows the non-buffering scenario where an audio/video stream is delivered straight from the network to the decoding/playout software. Figure 2(b) shows how buffering can be implemented by simply copying the incoming audio/video stream to a buffer while also delivering a copy to the decoding/playout software. The buffering software can be implemented as part of decoding/playout software or as a separate application. Figure 2(c) shows what happens when the user requests VCR functionality. The buffering software continues to copy the live stream to the buffer but the stream sent to the decoding/playout software is delivered from the existing buffer. Continuity can easily be provided by locally determining which frames the player is expecting.

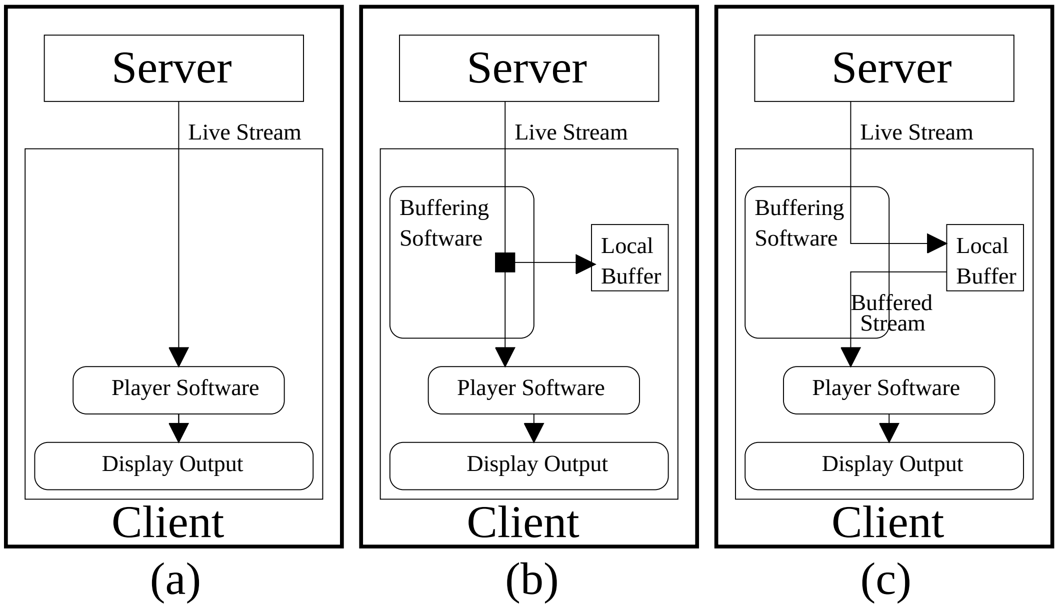


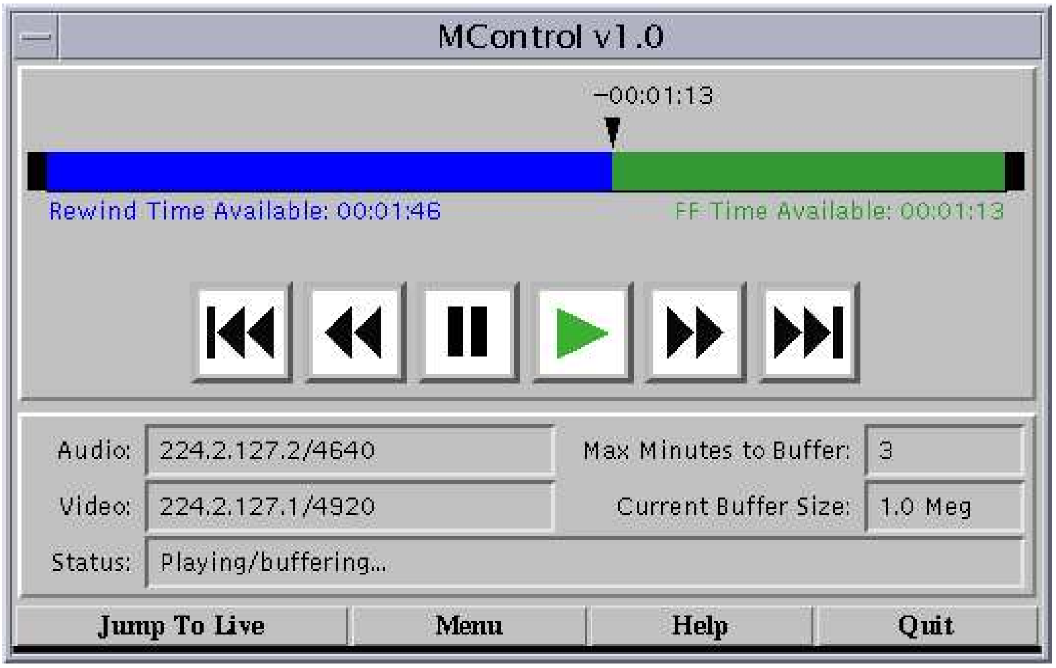
Figure 2: Client-side buffering.

The concept of client-side buffering is not new, but the challenge is to understand the costs associated with implementing it and to understand its effect on today's streaming multimedia environments. In particular, we focus on the implementation of client-side buffering for MBone sessions, both broadcasts and real-time conferences. Our implementation, called *MControl*, is a Java-based VCR interface providing client-side buffering for streaming audio/video. *MControl* is the buffering software and is built to work with the existing MBone tools for audio/video decoding and playout. In this section *MControl's* user interface design and implementation details are discussed.

4.1 User Interface

Figure 3 shows a snapshot of the *MControl* tool in use. From top-to-bottom, there are three parts to the interface. First, at the top of the interface, a meter bar indicates the user's playout location in the local buffer. Captions to the bottom left and right of the meter bar display the time in minutes and seconds available for rewind/pause or for fast forward functions. The various states of the buffer will be described in further detail in the next section, but some specific observations about this particular snapshot include:

- The pointer above the meter bar indicates that the user is watching 1 min 13 secs behind the playout point being sent by the server.
- The user can fast forward for up to 1 min 13 secs, at which point the user will be watching the most recently delivered frames.
- The user also has up to 1 min 46 secs available for rewind or can pause for up to 1 min 46 secs. If the user were to rewind or pause for this duration, the buffer would contain 3 minutes of unwatched data and *MControl* will not be able to support additional pause or rewind functions.
- The user has been watching for at least 3 minutes because there is 3 minutes of data in the buffer. If the user had been watching for less than 3 minutes, the buffer would not be full yet. The pointer above the meter bar is stationary because the buffer is in equilibrium, i.e. frames are being discarded at the same rate they are being received.



Rewind Time Available: 00:01:46
FF Time Available: 00:01:13

Audio: 224.2.127.2/4640
Video: 224.2.127.1/4920
Status: Playing/buffering...
Max Minutes to Buffer: 3
Current Buffer Size: 1.0 Meg

Figure 3: The *MControl* user interface.

Second, below the meter bar, there is a set of standard VCR-style controls. The pictorial representations from left to right are: *rewind to beginning*, *rewind*, *pause*, *play*, *fast forward*, and *fast forward to end*. The buttons implement intuitive VCR-style functions. The *rewind to beginning* button moves playout to the beginning of the local buffer and the *fast forward to end* button moves switches the playout from the buffered stream to the live stream. In addition to the VCR buttons, complete random access within the buffer is achieved by clicking and dragging the meter bar pointer to the desired playout location.

Third, below the interactive function buttons, there is a set of status fields and operational buttons. The multicast IP address and port of both the audio and video session are displayed. A "Minutes in Buffer" field displays the number of minutes of streaming media the local buffer is configured to store. The "File Size" field shows the current size of the local buffer in megabytes. A "Status" field displays messages to the user. Four buttons line the bottom of the screen. "Jump to Live" is the logical equivalent of the *fast forward to end* button, and allows a user to switch from the local buffer to the current live stream. The "Menu" button brings up a configuration screen which allows the user to specify the session address and buffer size. The "Help" button brings up a window of help text and the "Quit" button exits the application.

4.1.1 Representing the Buffer

Local buffering does not provide full interactivity all the time, especially when there is a system constraint that all interactivity must be handled locally. There are two specific limitations which manifest themselves in the particular type of client-side buffering implemented in *MControl.* They are:

1. **Starting Condition**: When the multicast group is first joined, the buffer is empty. The buffer will fill as streamed data is received. During this initial period, the duration of the rewind function is severely limited. Because there is no data in the buffer, there are no frames available for rewind. However, the pause function is fully functional (up to the limits of the buffer) because pause only stops playout and does require old frames to be replayed. Finally, because the user is watching the most recently delivered frames, fast forward is not possible.

2. **Buffer Size Limitations**: The size of the buffer is another constraint on the VCR functionality. Although one of the fundamental principles of client-side buffering is that disk space is cheap, available buffering will always be finite. In the case when high bandwidth or long duration programs are buffered, the buffer may only be capable of holding a few minutes of data[1]. When the buffer becomes full, the oldest data must be replaced by arriving data. As frames are buffered, two states are possible: the buffer can either be filling or it can be full. The implications of each of these states are:

    - *Partially Filled Buffer*: When the buffer is not full, the start of the buffer is fixed and the incoming streaming data is appended to the end of the buffer. Equilibrium has not yet been achieved because all incoming frames are being stored and nothing is being discarded. During this period, user interactivity is limited to movement among frames that exist in the buffer.
    - *Completely Full Buffer*: When the buffer fills, *MControl* must begin discarding the oldest frames in order to make room for arriving frames. An equilibrium between discarded and

[1]The buffer size restriction is specified in time rather than in disk space, primarily because variable rate streams would make it almost impossible to accurately represent how long the remaining buffer space would last. Time as the unit of buffer space is also a more meaningful statistic from a user perspective.

arriving frames is established. The user is still limited to interactive movement among the frames in the buffer but the total range of movement is only limited to the total size of the buffer.

Conceptually, *MControl* implements a "circular buffer". The various buffer states and the location of the playout pointer affect the ability of *MControl* to provide full interactivity. The specific interactivity limitations vary by VCR function and can be broken down as follows:

- **Pause**: The cumulative duration of all pause actions may not last longer than the size of the buffer. However, pause time can be "recovered" if the user initiates a fast forward action. A user attempting to pause longer than the buffer duration would interrupt playout continuity. In this case, the user's playout point is stationary while new frames continue to arrive. The number of frames between the user's playout point and the live playout point would be too great to store in the buffer. The best approach to handle this situation is to force the user to resume playing. Since playout and the incoming frames should be at a relatively equal rate, the buffer will be sufficient to insure continuous playout. If a user now initiates a fast forward action, the user's playout point will catch up to the live playout point. In this case, the user is recovering additional buffer space which can be used for future pause actions.
- **Rewind**: Pause and rewind are actually very similar actions from the perspective of managing the buffer. With pause, the user's playout point is stationary but the distance to the live playout point is increasing. With rewind, the user's playout point is moving backwards but, like pause, the difference to the live playout point is also increasing. A user request to rewind beyond the limits of the buffer is not possible because frames that need to played have already been discarded. The best approach in this situation is to rewind as far as possible and then resume normal playout. Again, if the user initiates a fast forward action, pause/rewind time will be recovered.
- **Fast Forward**: A fast forward action may take place at any time except when the user's playout point is equal to the live playout point. In other words, the user cannot fast forward beyond the current live playout point.

These limits on the user's interactivity can be confusing if the underlying principles are not well understood. The best way to help a user understand these limitations is to use an intuitive visual representation. *MControl* attempts to clearly visualize the user's limited random access capabilities by representing the buffer as a horizontal, time-based slider. A slider control is a familiar tool for many streaming media players. The width of the slider control traditionally represents the total duration of the media clip. A knob or pointer moves from left to right within the slider control as the clip plays, completing when it reaches the rightmost location. The user can click and drag the pointer to any location within the slider control for random access (see slider control in Figure 4).

For a slider control to work, the tool must know the duration of the media clip in advance. When a media clip is live, there is no way to know the total duration *a priori*. Indeed, most commercial

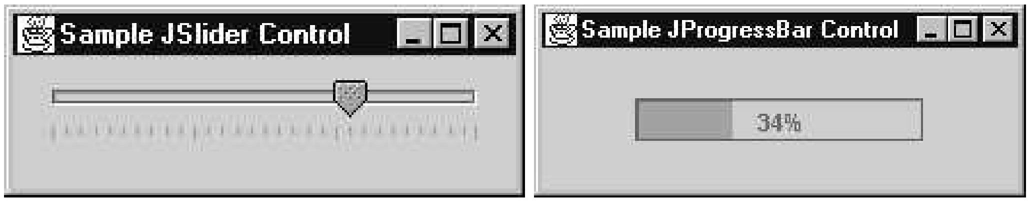


Figure 4: Simple slider and progress bar implemented as Java components.

media players do not show a slider when playing a live clip, or simply fix the pointer to the left side of slider control and do not allow it to be repositioned. When there is no predetermined duration for a media clip, clearly the slider control has limited use.

In *MControl* we have modified the slider control to be useful for streaming audio/video even for live events. Local buffering requires the display of two crucial pieces of information. First, the size of the buffer, and second, the location of the user's playout point in the buffer. Together, this information can be used to represent the state of the buffer and whether the user's playout point is changing with respect to the live playout point. *MControl*'s meter bar, shown in Figure 5, combines the functionality of a slider bar with a progress bar (see Figure 4). The modified meter bar is capable of showing the playout status, buffer status and VCR functionality availability.

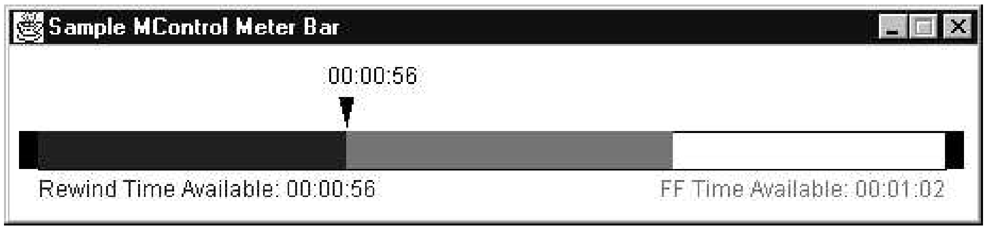
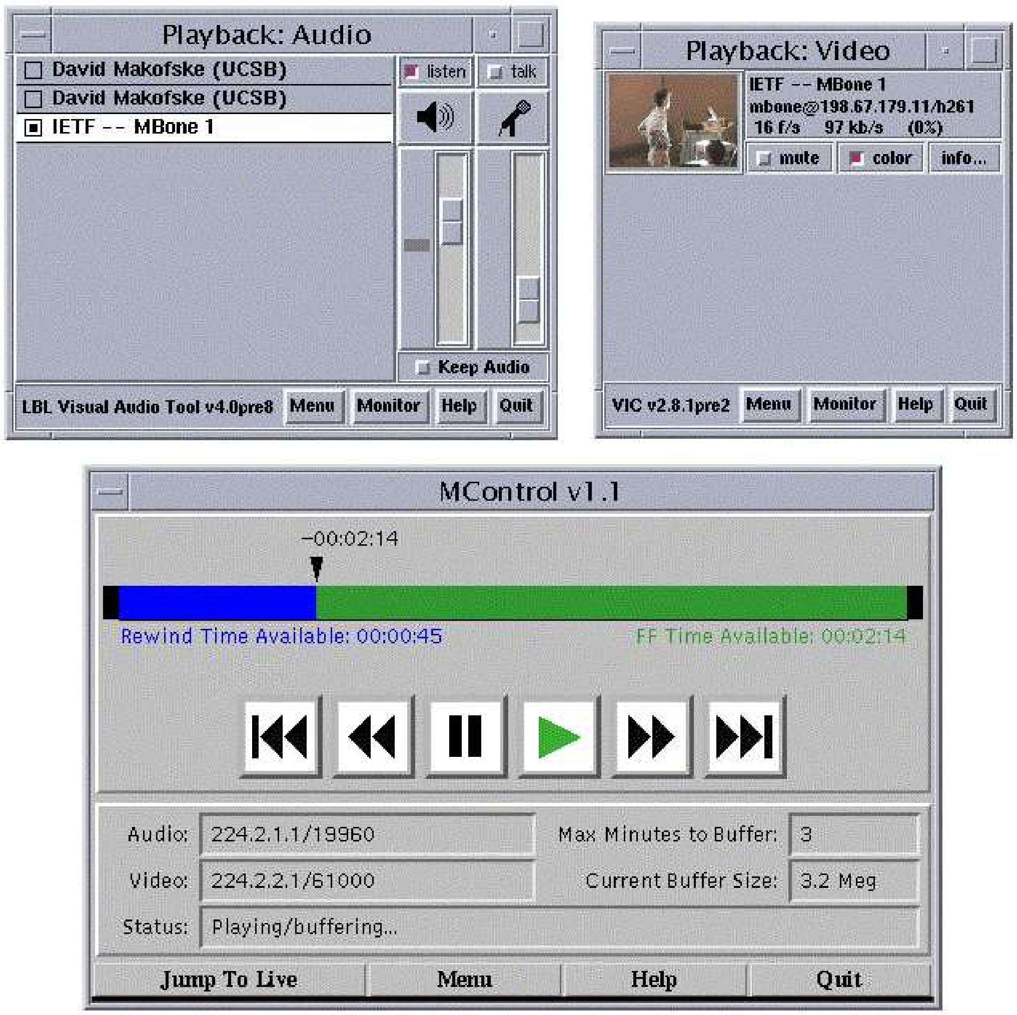

Sample MControl Meter Bar
00:00:56
Rewind Time Available: 00:00:56
FF Time Available: 00:01:02

Figure 5: The custom *MControl* meter bar.

4.1.2 Integration with MBone Tools

The *MControl* design is generic enough to work for any one-to-many streaming session, but the implementation was built to work specifically for MBone broadcasts and multi-party conferences. Because *MControl* is a prototype, one of the implementation goals was to use existing software as much as possible. We accomplished this goal for two major components: stream buffering and decoding/playout. *MControl* then only has to provide a user interface and stream control functions. Buffering is considered an internal component and is described in the next section.

Decoding and playout functions for *MControl* are provided using the audio and video tools most often associated with the MBone. When multicast was first implemented in the Internet, it was deployed as a virtual testbed overlaying the Internet[16]. The paradigm for creating sessions has been to use a session announcement tool, called the session directory tool (*sdr*), to periodically send session announcements on the MBone. MBone participants start *sdr*; hear announcements from various sources; and collect them in their session directory. If a user wants to join a session, *sdr* is able to start the audio and video tools with the proper options. For much of the MBone's existence, the tools most commonly used have been the audio tool *vat*, and the video tool *vic*[2]. *Vat* and *vic* are pictured with *MControl* in Figure 6.



David Makofske (UCSB)
David Makofske (UCSB)
IETF -- MBone 1

IETF -- MBone 1
mbone@198.67.179.11/h261
16 f/s 97 kb/s (0%)

-00:02:14
Rewind Time Available: 00:00:45
FF Time Available: 00:02:14

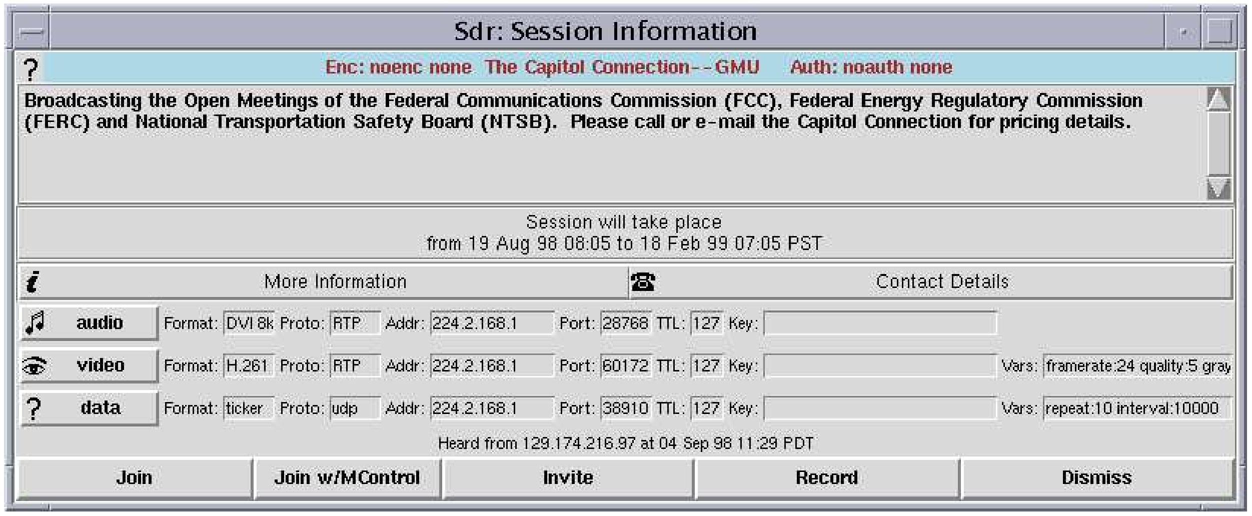
Audio:	224.2.1.1/19960	Max Minutes to Buffer:	3
Video:	224.2.2.1/61000	Current Buffer Size:	3.2 Meg
Status:	Playing/buffering...		

Figure 6: *MControl* with the *vic* video tool and *vat* audio tool.

One key to making *MControl* easy to use is the ability to incorporate it as a seamless component of the MBone tools. Because of the way *sdr* was designed, it has hooks to support customization for new applications. In the ideal situation, when a user starts a session, the *MControl* tool should be started along with *vic* and *vat*. This functionality was possible to implement with *sdr* by writing a tcl/tk script to add a button to the *sdr* session launch window. Figure 7 shows a snapshot of an

[2]Several sites offer a repository of MBone tools. One is http://mice.ed.ac.uk/mice/.

sdr window with this functionality. A "Start with MControl" button has been added to the set of possible user actions. When selected, this button allows the user to automatically start *MControl* along with the appropriate audio and video tools. Once started, the received stream will be buffered by *MControl* and displayed by the decoding/playout tools.



Enc: noenc none The Capitol Connection--GMU Auth: noauth none

Broadcasting the Open Meetings of the Federal Communications Commission (FCC), Federal Energy Regulatory Commission (FERC) and National Transportation Safety Board (NTSB). Please call or e-mail the Capitol Connection for pricing details.

Session will take place
from 19 Aug 98 08:05 to 18 Feb 99 07:05 PST

audio Format: DVI 8k Proto: RTP Addr: 224.2.168.1 Port: 28768 TTL: 127 Key:

video Format: H.261 Proto: RTP Addr: 224.2.168.1 Port: 60172 TTL: 127 Key: Vars: framerate:24 quality:5 gray

data Format: ticker Proto: udp Addr: 224.2.168.1 Port: 38910 TTL: 127 Key: Vars: repeat:10 interval:10000

Heard from 129.174.216.97 at 04 Sep 98 11:29 PDT

Figure 7: *Sdr* join window with an option for *MControl*.

4.2 Implementation

MControl has been written in Java and is a Java application. *MControl* has been designed to be cross-platform and is compatible with most Unix variants where the necessary MBone tools are available. The discussion of the implementation is divided into two parts: a discussion of the main program logic, and details about the capture and playout functions.

4.2.1 Main Logic

The *MControl* architecture is illustrated in Figure 8. The "mcontrol class" provides the graphical user interface (GUI), main program logic, and communication/control between the other program components. The three components controlled by the "mcontrol class" are as follows:

- The "Timer thread" provides callbacks to the main program every second to update the pointer location on the meter bar and perform other periodic, time-based events.
- The "DumpManager class" is responsible for all aspects of storage and maintenance of the local buffer.
- "The PlayManager class" is responsible for playing frames from the local buffer.

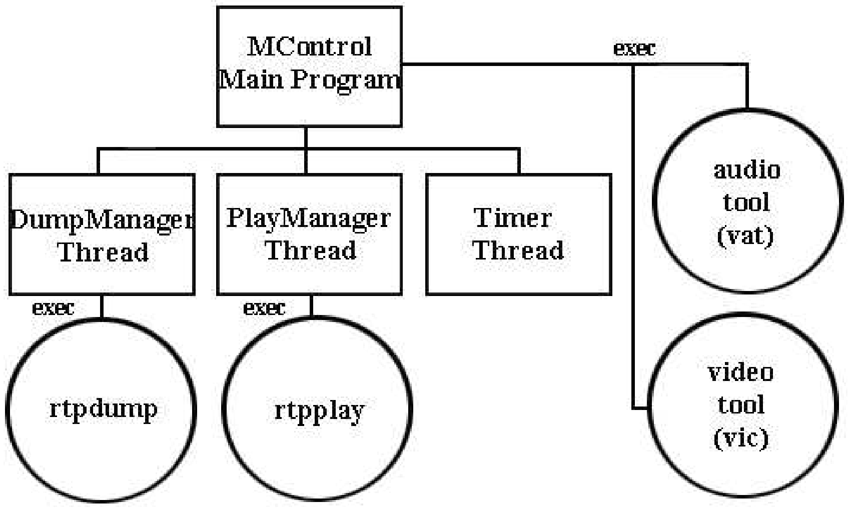


Figure 8: Overview of the *MControl* architecture.

The key functionality of *MControl* is to control the flow of frames as they arrive at the user's machine. Depending on whether the user is in "Live Mode" or "Playback Mode", *MControl* may have to send frames to the user from the network or from the local buffer. When a user is playing frames from the local buffer, the "PlayManager" is active.

The functions of frame buffering and local playout are accomplished using a set of multicast addresses and ports different than the live session's addresses and ports. When a user switches from "Live Mode" to "Playback Mode" the "PlayManager" starts playing buffered frames to a loopback or local address. In conjunction, the decoding/playout software must be "re-tuned" to this loopback or local address. In this way, frames arriving from the server will not interfere with frames being played from the buffer. While the user is in "Playback Mode", the "DumpManager" continues to capture packets from the live group.

Because we made the decision to use *vic* and *vat* without making any modifications, there were certain problems which had to be overcome using "non-optimal" techniques. One example is based on the limitation that *vic* and *vat* cannot be re-tuned to a new multicast address. Therefore, when a user switches from "Live Mode" to "Playback Mode" and from "Playback Mode" to "Live Mode", the *vic* and *vat* processes must be killed and a new instantiation of the tools started.

4.2.2 Capture and Playback Functions

For ease of implementation, we have utilized an existing set of utilities to store and replay audio/video data. These tools, collectively known as the RTPtools[3], are used to capture and replay data encapsulated in the Real-time Transport Protocol (RTP)[17]. Developed at Columbia Univer-

[3]Available from ftp://ftp.cs.columbia.edu/pub/schulzrinne/rtptools/.

sity, the RTPtools suite provides a number of individual tools providing different functions. The *rtpdump* tool records incoming RTP packets from a specified address and stores them in a file. The *rtpplay* tool reads *rtpdump*-generated files and re-creates the RTP packet stream on an outgoing address. The implementation of stream capture and playout functions are based on these two tools. Each function has been implemented as two separate Java classes: the "DumpManager" and the "PlayManager".

DumpManager. The "DumpManager" runs as a separate thread and takes as input: (1) the source IP address and port of each media type, and (2) the maximum buffer size. One problem with using the two *RTPtools* is that they can only record and play packets to files. This somewhat complicated the implementation of the *MControl* "circular buffer". Ideally, when the buffer is full, *MControl* should drop packets off the tail of the buffer at the same rate as they are being added. However, the *RTPtools* cannot manipulate packets, only files. To solve this problem, the buffer is broken into multiple fixed-duration files. When *MControl* is started, it buffers data in 30 second chunks with each chunk stored as a separate file. *MControl* can provide continuous playout even across file boundaries. The extra overhead of this techniques is incurred when data becomes inaccessible to the user but is still in a buffered file. Instead of deleting packets off the end of the buffer, *MControl* deletes data a file/chunk at a time. Consequently, some packets are stored for an extra period of time. In the case of *MControl* unusable packets are kept up to 30 seconds, the length of time each file/chunk can hold. This process is shown in Figure 9.

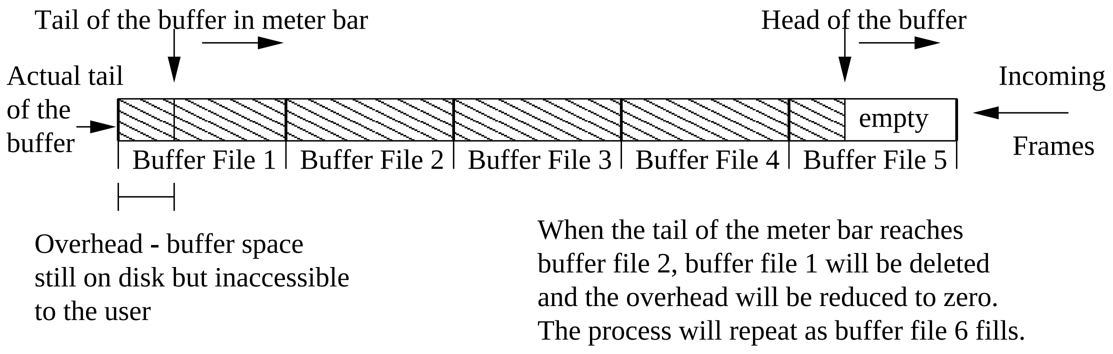

Tail of the buffer in meter bar
Head of the buffer
Actual tail of the buffer
Incoming Frames
empty
Buffer File 1
Buffer File 2
Buffer File 3
Buffer File 4
Buffer File 5
Overhead - buffer space still on disk but inaccessible to the user
When the tail of the meter bar reaches buffer file 2, buffer file 1 will be deleted and the overhead will be reduced to zero. The process will repeat as buffer file 6 fills.

Figure 9: *MControl* uses fixed-duration buffers to implement its circular buffer.

PlayManager. The "PlayManager" also runs as a separate thread. It is started from the main program in response to a user request which requires playout from the local buffer. The "Play-Manager" takes as input: (1) the destination IP address and port of each media type, and (2) the offset (in seconds) from the beginning of the buffer at which to start playout. The destination IP address and port can either be the local machine's loopback address or a randomly generated multicast address. The reason for using a different address is to ensure that the locally buffered stream does not interfere or overlap with the incoming live stream. If an address other than the loopback address is used, the time-to-live (TTL) field is set to 1. This limits the scope of the playout to the local network.

5 Results and Implications

Despite the relative complexity of representing client-side buffering to the end-user, *MControl* is a fairly intuitive and easy-to-use application. The traditional VCR-style interface is familiar to many users, and the *MControl* meter bar clearly represents the bounds of the user's interactivity. The authors and a high percentage of users who were formally surveyed found *MControl* to be a valuable tool for watching live conferences and lectures.

The availability of a tool like *MControl* raises several interesting questions about how the tool and its functionality will affect user behavior in certain kinds of sessions. One of the most interesting applications is real-time, multi-party conferencing. Specific applications include meetings, distance learning or Internet-based teleconferences. Giving users the ability to rewind and replay an important discussion will likely be invaluable for improving the utility of distributed meetings. However, interesting problems arise when one user wishes to interact with another user. The potential exists for the second user to not actually be watching the live stream, but rather, is watching from the buffer. The fundamental problem is how to deal with temporally displaced participants within a single, interactive session. Currently, there are no mechanisms to inform other session participants that a user has entered their local buffer and will not be able to immediately respond to questions.

Not only is *MControl* functionality a problem for session coordination, but it impacts the semantics of state information carried by the Real-time Transport Control Protocol (RTCP). RTCP is part of the RTP standard and its function is to carry group participant and reception quality information on a separate control channel. Each member of a session sends periodic RTCP packets to all other group members. As a consequence, a participant will receive packets from other group members. These packets are used by each member to establish a group list and learn about other members' reception quality.

MControl challenges some assumptions made by RTCP, and can invalidate the group information displayed by most decoding/playout tools. Three different problems can be created. They are:

- **Ignored Members**: *MControl* may cause existing group members, who should appear in the group list, not to be displayed.
- **Indirect Members**: *MControl* may show group members in the session list, but who are not watching at the same playout point as other group members.
- **Ghost Members**: *MControl* may show group members who have actually left the group and are no longer participating.

A scenario in which several of these problems might occur is when a user enters "Playback Mode". The *vic* and *vat* tools that were decoding/playing the live multicast stream are killed and replaced with versions tuned to the local buffer playout. Two problems occur. First, when the second set of tools is started, there is no participant information because the original list was deleted when the first set of tools was killed (ignored members). Second, even though the user who is running *MControl* is still listening to the session (but now at a time offset), RTCP packets are no longer being sent to the live group. This gives the appearance that the user is no longer part of the group (indirect member).

Another scenario with problems is caused by the semantics of *rtpdump* and *rtpplay*. These tools record and play *both* RTP and RTCP packets. As a result, the locally displayed group membership list may become corrupted. For example, if a user is in "Playback Mode", the membership list will be built from the RTCP packets stored in the buffer. This represents not the current group membership, but the group at the time the data was buffered (indirect and ghost members). Further complicating the problem is what happens when a user initiates a VCR action. Suppose a user fast forwards for a time and then begins to play again. If, during the fast forward period, the user skipped over another user leaving the group, *vat* will have missed the "BYE" packet. When play resumes, the user who left will still appear to be in the group (ghost member). A pictorial representation of this problem is shown in Figure 10. The initial group members are users A and B. After fast forwarding, users A and B have left and user C has joined, but the local user missed the two leave and one join actions. A and B are ghost members and C is an ignored member. Heavy use of the the VCR functions can create group lists very different from any group composition that ever actually existed.

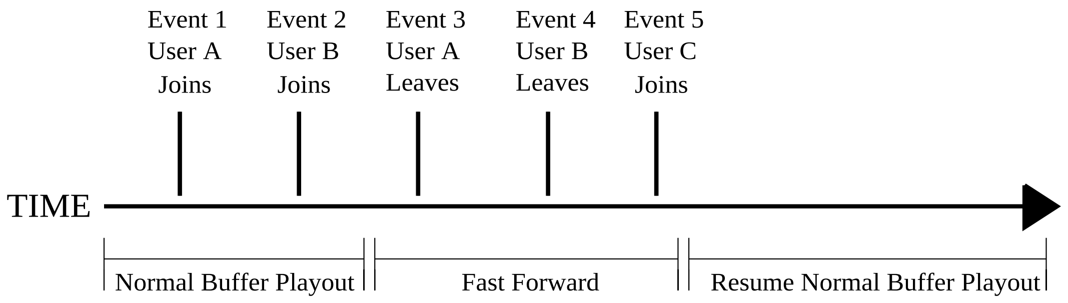


Figure 10: How interactive functions can corrupt RTCP control data.

There are a number of possible solutions to the problem of accurately tracking group membership. However, most solutions either require additional overhead; changes to the decoding/playout tools; or changes to the RTCP protocol. Anything more than the consideration of possible solutions is beyond the scope of this paper. Possible solutions include:

- When in "Playback Mode", *MControl* could continue to send RTCP packets to the live session with a text "disclaimer" indicating that the user is accessing the session from a local buffer, and is not currently listening to the live stream. A more precise solution would be to include the actual "offset from live" time in the RTCP packet. Unfortunately, RTCP packets are not sent often enough to carry sufficient granularity.

- To prevent group list corruption amid frequent local playout modifications, RTCP packets in the buffered stream could be processed even though a user was pausing, rewinding, or fast forwarding the audio/video. *MControl* would be responsible for maintaining an accurate list of group members by monitoring the local user's playout point, buffered RTCP packets, and live RTCP packets.

6 Related and Future Work

MControl is one of the first attempts to provide VCR-style functionality for both live and stored video or audio streams delivered over a network using multicast communication. In most cases, existing multicast-based recording tools only allow a stream to be recorded and re-played at the conclusion of the recording period. These tools work in a manner similar to how many VCR's operate for television programs, i.e. a person can watch and record at the same time but cannot use VCR functions while recording. The MBone VCR[18] is one tool that provides this traditional functionality. An alternative is the MBone VCR-on-Demand (MVoD) tool[19] though it adds the ability to record and play multicast sessions remotely, facilitating the creation of a VoD service. The Multicast Multimedia On-Demand (mMOD) tool[20] performs similar VoD functionality but adds a WWW-based playout request mechanism and attempts to improve playout quality. None of these tools provide real-time interactive functionality for a live set of multimedia streams.

Given the existing tools and the additional functionality provided by *MControl*, several directions are possible for enhancing functionality even further. The ability to record the live stream would complete the VCR analogy and allow complete random and unrestricted access even after the live transmission is finished. The ability to "bookmark" a particular time in the stream so that it could be accessed at a later time would allow random access indexing similar to the MBone VCR[18]. Finally, the seamless provision of VCR-style interactivity could be improved if *MControl* were more closely integrated with the MBone tools themselves. Whether the best approach is to move *MControl* functionality into the MBone tools or vice-versa is left for future investigation.

7 Conclusions

The *MControl* prototype provides client-side VCR-style interactive functions for live and stored multicast streams, cases where interactivity has traditionally been difficult to provide. The challenge is to give participants in a multicast group the perception that they individually control the playout of the stream they are watching. Instead of relying on techniques which require server involvement in playout control, a central bottleneck already, we impose the entire responsibility on each client. While this limits a user to partial VCR functionality, the limitations are not particularly significant and can be well understood by a user simply by presenting an intuitive interface. This is accomplished with the *MControl* meter bar. In addition to designing and implementing *MControl*, we have attempted to understand some of the implications of using such a tool in interactive, multi-party environments. Informal qualitative analysis suggests *MControl* provides added value for users. The side effect of use in these environments is that users who seem to be participating and able to interact may actually be temporally displaced and not "in the present".

Since the release of *MControl* in 1998, a number of companies have released set-top boxes that allow similar functionality for traditional television[4]. However, there still has not been work by the WWW media player companies to incorporate this level of functionality into their tools. This is surprising especially considering the growing demand and interest in Internet-based programming. If convergence of computers and television is to take place, enhanced functionality will have to be implemented in production software.

References

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

[2] H. Linder, H. Clausen, and B. Collini-Nocker, "Satellite Internet service using DVB/MPEG-2 and multicast web caching," *IEEE Communications*, vol. 38, pp. 156–161, June 2000.

[3] 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.

[4] W. Mossberg, "VCR-challenged TV junkies find two firms offer options," *The Wall Street Journal Interactive Edition*, April 1999. Available at http://ptech.wsj.com/archive/ptech-19990408.html.

[5] D. LeGall, "MPEG: A video compression standard for multimedia applications," *Communications of the ACM*, pp. 47–58, April 1991.

[6] L. Rowe, B. Patel, B. Smith, and K. Liu, "MPEG video in software: Representation, transmission and playback," in *IS&T/SPIE 1994 International Symposium on Electronic Imaging: Science and Technology*, (San Jose, California, USA), February 1994.

[7] L. Rowe, B. Patel, and B. Smith, "Performance of a software MPEG video decoder," in *ACM Multimedia*, August 1993.

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

[9] A. Klemets, "The design and implementation of a media on demand system for WWW," in *World Wide Web Conference (WWW)*, (Geneva, SWITZERLAND), May 1994.

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

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

[12] K. Almeroth and M. Ammar, "An alternative paradigm for scalable on-demand applications: Evaluating and deploying the interactive multimedia jukebox," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 658–672, July/August 1999.

[13] A. Gelman, S. Halfin, and W. Willinger, "On buffer requirements for store-and-forward video on demand service circuits," in *IEEE Global Telecommunications Conference (GLOBECOM)*, (Tucson, Arizona, USA), December.

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

[15] X. Li, M. Ammar, and S. Paul, "Video multicast over the Internet," *IEEE Network*, April 1999.

[16] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment," *IEEE Network*, vol. 14, pp. 10–20, January/February 2000.

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

[18] W. Holfelder, "MBone VCR - video conference recording on the MBone," in *ACM Multimedia*, (San Francisco, California, USA), November 1995.

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

[20] P. Parnes, K. Synnes, and D. Schefstrom, *mMOD: the multicast Media-on-Demand system*, May 1997. Available at http://mates.cdt.luth.se/software/mMOD/paper/mMOD.ps.