The Design and Implementation of a Media on Demand System for WWW

Anders Klemets
Department of Teleinformatics,
The Royal Institute of Technology, Sweden
klemets@it.kth.se

Abstract

Current WWW clients do not directly support synchronized playback of continuos media. It is however possible to support continous media by using external programs that are controlled by the WWW client. This paper describes the design and implementation of a "Media on Demand" server which uses WWW as its user interface. Offered media include audio and video recordings of transmissions on the Internet Multicast backbone (MBONE), as well as arbitrary prerecorded audio files.

1.0 Introduction

A hypertext link in WWW may refer to an arbitrary object. Such objects might be text files, images, sound files or anything else. Most commonly, hypertext links point to documents written in HTML, the Hypertext Markup Language [HTML], which may contain other hypertext links. An HTML document is a representation of textual information. Apart from specifying hypertext links, the principal use of HTML is to format text and display characters using different fonts. The multimedia capabilities of HTML are quite limited. It is possible to include graphics in the flow of text, such as GIF images. This is called inlining. It is, however, not possible to inline arbitrary arbitrary multimedia objects, or even other textual documents.

An extention could be made to the inlining mechanism to make it possible to include audio files and MPEG movies in the flow of text. The playback of the movie or the audio file would begin when the text surrounding the inlined objects is displayed.

If an inline operation specifies multiple objects, or perhaps another HTML document, that in its turn inlines multiple objects, those objects should be played in parallel. This would make it possible to present a slideshow at the same time as a an audio file is being played.

It is still possible to use WWW to play parallel continous media streams, such as audio and video. But without some sort of extentions to HTML, such as those described above, the actual playback will have to be done by programs external to the WWW client.

This paper describes the architecture of one such system that uses audio and video tools that are being used for experimentation on the MBONE, the Internet Multicast Backbone.

2.0 Design considerations

A lot has been written recently in the press and in research forums about video on demand systems, see for instance [Rangan]. Some papers talk about systems that support storage and retrieval of HDTV quality (MPEG-2 encoding) movies, other settle for the more modest quality provided by MPEG-1 encoding of movies.

There are however three major obstacles that prevent the implementation of a video on demand system for MPEG movies.

2.1 Using MPEG movies

At 25 or 30 frames per second, an MPEG-1 encoded movie provides a quality similar to what might be obtained when watching a VHS movie recording. The corresponding transmission rate is 1.5 Mbit/s. This is not a very high speed for most local area networks. A 10 Mbit/s Ethernet should be able to support 4-5 concurrent transmissions. An FDDI ring or an 155 Mbit/s ATM network should be able to support anywhere between 20 and 100 concurrent transmissions. On a wide area network, it might however not be possible to support one single transmission. Many of the links in the current Internet have only a bandwidth of 1.5 Mbit/s or less.

Another obstacle is the storage size of movies. At 1.5 Mbit/s, one hour will require 675 Mbyte, which equals 1.35 Gbyte per feature length film. Storing a reasonable collection of films will require either large arrays of expensive disks, or some, possibly complicated, system based on tertiary storage such as tape stations. [Federighi]

Yet another problem is the processor speed. 1.5 Mbit/s corresponds to approximately 1.8 kByte/s. If data is sent as 1500 byte packets, which is the maximum permitted content of Ethernet packets, this will yields a packet intensity of 125 packets per second. That is to say, one packet will be sent every 8 ms. Depending on how long it takes the server to generate one packet, this may severely limit the number of concurrent transmissions that can be sustained.

Although dedicated routers should be able to support intense streams of packets, most multicast IP routers are currently UNIX workstations. There are reasons why it may be desirable to the use the multicast IP, which will be explained in the next section. Measurements have shown [Jacobsen] that Sun Sparcstations need about 1 ms to forward a multicast IP packet, per interface. If such a multicast router forwards each packet to 4 other routers, (not an uncommon configuration) it will thus only be able to sustain two concurrent video transmissions.

We wanted to implement a video on demand system that would be able to scale beyond one or two simultaenous transmissions. It should also be able to work with present computers on todays Internet. These requirements made it clear that we would have to settle with video quality inferior to what is provided by MPEG-1.

2.2 Using the Mbone tools

The MBONE is a logical network that is built on top of the Internet. It provides multicast IP [Deering] connectivity between several hundred LANs on the Internet. Multicast IP [RFC-1112] is an extention to the IP protocol that makes it possible to denote a group of recipients by one single IP address. Essentially, when an IP datagram is sent to such a multicast group, it will be delivered to all hosts that are members of that group.

The routing of multicast packets on the MBONE is implemented so that multicast packets are only duplicated when the path to group members branch at a router. When the transmission media of a link supports multicast, such as Ethernet, the IP multicast datagrams are transmitted using link level multicast.

The efficient routing of multicast packets on the MBONE has made it tractable to use it for experiments in multimedia conferencing. Such experiments have been conducted for about three years [Casner] using various tools for transmitting audio, video and whiteboard data.

Conferences or seminars are regularly being sent on the MBONE from Europe, USA and sometimes also Australia. Although remote conferencing makes it possible to participate in a conference from the convenience of ones own office, the time differences between Europe and the USA often make remote attendance impractical.

A reasonable solution to the problem would be to record the transmissions on a disk and play them back at a later date.

We implemented a series of tools to record and replay captured conferencing traffic and eventually realized that these recording tools, together with the normal conferencing software, were good candidates for use in a WWW based video on demand server. Since the conferencing tools are not limited to video, is more correct to talk about a media on demand server, as opposed to a server limited to video only.

There are two popular programs for transmitting video on the MBONE, `ivs` by Thierry Turletti at INRIA, and `nv` by Ron Frederick at Xerox PARC. Normally, these programs are used with a separate program for transmitting audio, `vat` by Van Jacobsen and Steve McCanne at LBL. A typical video transmission using nv requires approximately 128 kbit/s and the audio adds an extra 64 kbit/s. This is roughly an order of magnitude less than what is used by MPEG-1. The image is only a quarter of the size of the original video frame and the frame rate is usually less than 5 frames per second. This quality is not acceptable for transmitting feature length movies. But this is often sufficient for the purpose of transmitting images from a conference. As long as the sound is adequate, it might not be important to have full motion video of the speaker.

If the camera is pointed at the overhead display at the conference site, the low frame rate should be of even a lesser problem. Although contrast and resolution may still be a problem, it is recommended to use a distributed whiteboard program, (several such programs are available) to disseminate the slides.

In an effort to avoid large packetization delays which might be irritating during two-way communication, the vat and nv programs send fairly small size packets. This also helps to reduce the perceptible impact of lost packets. The drawback is, of course, that a higher packet intensity is imposed on the network. Nv generates about 40 packets per second, and vat generates 25 packets per second if transmitting 40 ms worth of audio in each packet. Using the same figures as above, a Sparcstation based router will receive one packet every 15 ms. If it makes 4 copies of each packet, it will be able sustain 3-4 concurrent transmissions.

3.0 The server implementation

The server uses NCSA HTTPD 1.0a5 running on a Sun Sparcstation. A number of recordings of MBONE transmissions are stored on a 1 Gbyte disk which is attached to a Sparcstation 10/30. All recordings include captured vat traffic, but only a select few also have nv video.

There are two distinct methods of accessing the recordings. The one that was implemented first uses the fill out form facility of HTML+ [HTML+], a proposed extention to HTML, that was first implemented in NCSA Mosaic 2.0.

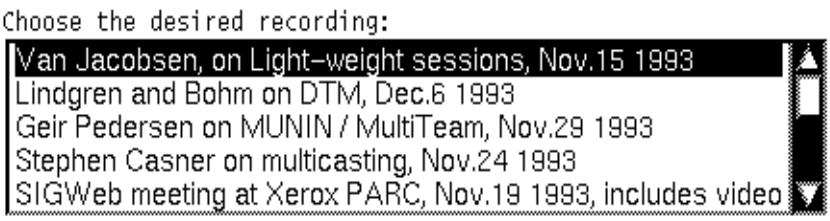

Figure 1.

3.1 Form based access

When accessing the form, the user is presented with a scrollbar menu listing different available recordings. See Figure 1 for an example of the recordings that are offered from the current form. In most cases, the only selection the user has to make is to choose the recording and to choose whether a video transmission is also desired. If the user thinks that the bandwidth between the server host and the recipient machine is low, it might be advantageous to disable the video transmission since it will decrease the probability of losing audio packets.

The fill out form also lets the user supply the destination address of the retransmission. Its default value is the hostname of the machine running the WWW client. This destination address might be a normal IP hostname or an IP multicast address, denoting a group of recipients. The recipient host need not be attached to the MBONE, but if multicast IP is to be used, there needs to be some sort of forwarding mechanism set up between the server and the recipient. The conferencing software can also be used on hosts that do not have IP multicast support, but in this case there is no support for multiuser conferencing. The scope of IP multicast packets in the media on demand server has been deliberately limited, due to problems that will be described in section 4.

It is also possible to choose the UDP ports being used for audio and video. The default values are usually adequate, but it is imperative that the UDP port values do not coincide with any UDP ports that are already in use on the recipient machine.

Upon submitting the form, a script will be executed at the server side to parse the current selections. The choice of recording will be presented to the script as a key/value pair. The value of the key will be the same textual string as the user was presented with in the scrollbar menu. The script must map a string such as “John Doe, Nov.19, 1993” into a the directory path of the recording and set a variable indicating whether this recording has a video component.

This unfortunately makes maintaining the library of recordings somewhat cumbersome. Adding a new recording requires modifying both the file containing the fill out form, and adding a mapping between the name and the directory to the server script.

It would be more convenient if the directory information could be contained in the form itself and be supplied as value to the server script. Recently, NCSA Mosaic 2.2 has been released, and it supports such a VALUE attribute to the <OPTION> tag in fill out forms.

The server script then proceeds to generate a custom HTML document to control the recording. This document includes references to how to start the vat and/or nv programs with the right parameters to be able to receive the transmission. The document also includes a link to a script that returns a document with the MIME type `application/x-csh`. This makes it possible for clients to have the multimedia programs started automatically, if their WWW client is configured to recognize such documents.

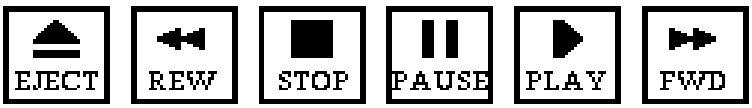

Figure 2.

At the bottom of the control page there is a set inlined bitmaps representing the controls of a tape deck, ie. start, stop, pause, forward and rewind buttons. See Figure 2.

Pressing the play button will execute a script at the server that starts the `vat_play` (for audio only) or `av_play` (for audio and video) programs if they are not already running. The script will execute a `rsh` command to the host that has the disk with the recordings, rather than executing playback programs directly on the WWW server host.

When the other buttons are pressed, or the play button is pressed after the playback has been started, the script will execute a remote control program. This program, `playremote`, will identify the correct instances of the playback programs and pass the remote control command on to them.

The server also has a mechanism for logging requests and it keeps track of how many recordings are currently being played. In our current configuration it will only accept 5 concurrent retransmissions in an attempt to limit the load on our local area network.

3.2 Direct access

After using the fill out form for a while, we realized that in most cases, the default values need not be changed. If the server knew which recording the user wanted, it could generate the control document directly, using the default values.

The solution was to have one single hypertext link describe the desired recording. It could then be embedded in textual documents describing the recordings.

Using this method, it is now possible to put the program of a conference on line, and have hypertext links at the text describing each presentation. Following one of those links would give the user the playback control page directly, with suitable default values.

The URL of such a self contained hypertext link can look like this `http://server/htbin/playdefault/conf1/talk1?Johns+speech`
The server would execute the script `htbin/playdefault` and it would generate the control document for the recording contained in the directory `conf1/talk1`. The string "Johns speech" would be used as the title of the control document.

This method has the advantage that was not originally possible using form based access, namely that new recordings can be added just by modifying a HTML document. The server script need no longer be modified.

It is still possible that the user may want to use other settings than the default values. The control page provides the user with a link to a fill out form which makes it possible to change the chosen parameters. This form is dynamically generated, and it does not allow the user to choose between different recordings. In the example above, the form will look like the original fill out form, but the only recording offered will be "Johns speech".

The recording is offered not as a scrollable list, but as a *radio* button. The reason for this is that radio buttons take a `VALUE` attribute, which makes it possible to supply the directory path ("conf1/talk1") as a value instead of the descriptive string. When the form is submitted, the server script will need to get the directory path from somewhere. The only place it can get this information from is from the values provided by the form.

It might look somewhat awkward to have a radio button with only one selection, but it used to be that all values supplied by a form had to be visible by the user. A possible workaround would be to supply the directory path information as an extention to the pathname that specifies the script to be executed when the form is submitted. But this should not be necessary as NCSA Mosaic 2.2 now supports invisible `INPUT` keys/value pairs if they are marked as having the type `HIDDEN`.

3.3 Problems with the user interface

The interaction between the WWW clients and servers is handled by HTTP, the HyperText Transfer Protocol. It became clear that HTTP was not designed for remote control operations. In particular, when the WWW client refers to a hypertext link, it always expects another document in return from the server. The server may execute a script or look up something in a database as a result of a HTTP request, but it must always return something.

When using hypertext links to remotely control the playback of a recording, the client should remain at the control document. It would be very annoying if it would load a new HTML document every time, say, the "forward" button is pressed. One of the disadvantages would be that it would generate a trace for each button that was pressed. If a user of NCSA Mosaic wanted to return to the document that preceeded the playback control document, by pressing the "Back" button, he would first have step through the documents generated by every press to any of the remote control buttons.

Fortunately, NCSA Mosaic provides a solution to this problem. It recognizes a non-standard message from the server, <mosaic-access-override>. Upon receipt of such a message, the client will consider the transfer complete, and remain showing its present document. NCSA Mosaic 2.1 now supports an official HTTP/1.0 return message, "204 NoResponse", that has the same effect.

There are still other problems with HTTP that makes the tape deck metaphor less than adequate for remote control of the playback. In particular, HTTP provides no information about for how long the mouse button was pressed while pointing at the hypertext link. This makes it difficult to provide continous forward or rewind operations. Rather, each press at the forward button will advance the recording by a fixed amount, such as 60 seconds. It does not matter for how long the button was pressed. Since there is often some latency between the press of the button and the receipt of the remote control command at the server, it is tempting

for the user to press the forward button several times. This is especially true if the user wants to skip forward significantly in the recording. But Mosaic does not allow one to queue multiple requests, so pressing buttons while the server is in the midst of serving a request has no effect.

It is not certain that adding the ability of recognizing multiple requests or the ability to submit aggregate requests to HTTP is a good idea. One must draw the limit somewhere, otherwise there is a risk that the clients and/or HTTP becomes overly complicated. An alternative approach would be to design a different user interface that makes better use of available functionality.

Figure 3.

An example of such a user interface is shown in Figure 3. There are two horizontal arrows with long shafts pointing in opposite directions. When the user points and clicks at an arrow, the amount of time to skip forward or rewind would be calculated from the offset of the pointer to the base of the shaft. This can be done by using the image mapping techniques that are already built in to NCSA Mosaic and many HTTP servers.

4.0 Multicast for group communication

The access metaphor provided by our media on demand system and other proposed video on demand systems, is similar to the one offered by video rental stores. The user rents a movie and is in full control of the playback of the movie. The film may be started and suspended at the users leisure.

The server will start to retransmit the recording immediately, irrespective of other replays that may already be in progress. Full control lies with the user of the WWW client. A malicious or unexperienced user may start several parallel retransmissions. Eventually the server will refuse to start other transmissions in order to preserve network resources. Since each recording typically lasts one or two hours, starting multiple transmissions is a simple denial of service attack for malicious users.

There is however a way of detecting if somebody is still receiving the transmissions. The recorded media is transmitted using the UDP protocol to a specific UDP port. If there is no program on the destination host that is prepared to receive UDP packets that are addressed to that particular port, the host will return ICMP Port Unreachable messages to the sender. When the server has received a number of such ICMP error messages, it will conclude that there is nobody receiving the packets, and abort the transmission.

This method is not failproof, however. If the network is overwhelmed by the amount of traffic, either the UDP packets or the ICMP packets may not reach their intended destination.

Transmitting the UDP packets using multicast IP imposes additional problems. Hosts are not allowed to send ICMP packets as a response to a multicasted packet under any circumstances.

The only way to end a transmission is for the user who started it to use the buttons in the WWW control page.

The problem of malicious or accidental transmissions of recordings that nobody wants to receive is further complicated in the multicast case. Different solutions have been suggested, that all make use of special ID messages sent by the conferencing programs. Vat sends these ID messages to the multicast group, while nv sends its ID messages using unicast IP to the sender of the video images. The idea is that the server should terminate the transmission if it detects the absence of these messages.

One possible solution is for the server to listen for ID messages from any group member. It aborts the transmission if it does not receive any ID messages after a given period of time. Unfortunately, usage patterns on the MBONE seem to indicate that users have a tendency to "tune in" on multicast groups without actually being interested in the traffic. Often users join a multicast group, and then leave their offices while still remaining members of the group. Thus, this scheme would lead the server to incorrectly believe that there still are users who are interested in receiving the transmission, when there in fact might not be.

Another possible solution is for the server to listen for only those ID messages that originate from the user that originally requested the transmission. This method has a problem opposite to the scheme described earlier. If there are other users who are in fact interested in receiving the transmissions, these users will be cut off if first user leaves the group. This is particularly prone to happen when retransmission is requested from an office workstation, but the recording is to be viewed at a different location such as conference room.

Both methods also suffer from the complication that the ID messages generated by vat are being sent using IP multicast. The MBONE sometimes experiences outages due to malfunctioning multicast routers. These outages can last several minutes and may erroneously cause the server to abort its transmissions. Since the underlying unicast network, the Internet, is still functioning, this could have been avoided had the ID messages been sent directly, with unicast, to the source.

Even when disregarding these problems, which will not occur as long as the users are well behaved, multicast communication poses other problems. The original access metaphor is altered to something that resembels cable TV "Pay Per View" systems. The question now is who should have the remote control of the playback. If the unicast communication model is simply extended, it will be the user who first requested the transmission, but control can also be shared by all members of the multicast group. This latter approach is not likely to be adequate unless the group is a small group of collaborating individuals.

If the transmission begins at the request of the first user, this may not satisfactory to the other users. When they join the group they may already have missed a significant part of the transmission. A different approach would be to assign a multicast group at the request of the first user, but schedule the transmission for a later time. This increases the possibility of multiple users being able to join the multicast group in time to receive the transmission from the beginning. From the perspective of saving bandwidth, it is desirable to schedule the transmission far in the future, so as to maximize the number of users that can follow the transmission. But this may not be satisfactory to the first user since the latency between the request and the beginning of the transmission may be very large. Applying this scheme makes it even more similar to cable TV "Pay Per View" systems. Much of the interactivity associated with video on demand systems is lost.

5.0 Implementation status

Our current media on demand server implements all of the features described so far. When the destination address is an IP multicast address, however, it does not attempt to determine whether there are any active receivers by listening to session ID messages. The control of the retransmission is left with the user who requested the retransmission. To reduce the impact of orphan retransmissions, the scope of the multicast packets are limited to our campus.

The part of the server that interfaces to WWW consists of more than half a dozen executables, Bourne shell and C-shell scripts and C-programs, plus a couple of additional HTML files. This makes it a little bit difficult to install and maintain. But for those who still want to look at the code, it is freely available with anonymous ftp from `sics.se`. The filename is `archive/media_on_demand.tar.Z`. The server scripts assume that the HTTP server is NCSA HTTPD 1.0a5 or later.

There is work in progress [Kumar] to rewrite the server in Perl. The Perl version should be much easier to maintain.

In addition, it is necessary to install the MBONE playback programs, `vat_play`, `nv_play`, `av_play` and `vat_play_audio`. The latter program allows one to send a normal Sun audio file as a sequence of vat network packets. The programs require the installation of IP multicast to the workstation to compile properly. But it should be possible to compile the programs on a system that does not support IP multicast with some minor modifications. The programs have been tested only on Sun Sparcstations, and might not necessarily compile on other systems. The programs are available for noncommercial use with anonymous ftp from `sics.se`. The filename is `archive/vat_nv_record.tar.Z`.

The users will need the `vat` program in order to receive the audio transmissions. It is available with anonymous ftp from `ftp.ee.lbl.gov` in the `conferencing/vat` directory. The `nv` program, which is used to receive the video, is available with anonymous ftp from `parcftp.xerox.com` in the directory `pub/net-research`.

6.0 Summary

We have described the design and implementation of a server that allows users to request retransmissions of audio and or video recordings. These recordings typically originate from the MBONE, but it is also possible to request audio files. The media will be delivered from the server to one or multiple recipients using conferencing tools used for experimentation on the MBONE. We have identified some problems that occur when one tries to apply the “tape deck”, “video on demand” and “pay per view” metaphors to a system like ours.

7.0 References

[Casner] S. Casner, S. Deering, *First IETF Internet Audiocast*, ConneXions, No.6:10-17, June 1992

[Deering] S. Deering, *Multicast Routing in a Datagram Internetwork*, Ph.D. Thesis, Stanford University, 1991.

[Federighi] C. Federighi, L.A. Rowe, *A Distributed Hierarchical Storage Manager for a Video-on-Demand System*, Storage and Retrieval for Image and Video Databases II, IS&T/SPIE, Symp. om Elec. Imaging Sci. & Tech., San Jose, CA, February 1994.

[HTML] *HyperText Markup Language specification*, work in progress.

[HTML+] *HTML+ HyperText Markup Language specification*, work in progress.

[Jacobsen] V. Jacobsen, message posted to `rem-conf` mailing list, June 1993.

[Kumar] V. Kumar, personal communication, 1994.

[Rangan] P. Venkat Rangan, H.M. Vin, S. Ramanathan, *Designing an On-Demand Multimedia Service*, IEEE Communications Magazine, Vol. 30, No. 7, July 1992.

[RFC-1112] S. Deering, *Host Extentions for IP Multicasting*, Internet Request for Comments no. 1112, August 1989.