

Real Time Video and Audio in the World Wide Web

Zhigang Chen See-Mong Tan Roy H. Campbell Yongcheng Li
Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield
Urbana, IL 61801
{*zchen,stan,roy,ycli*}@cs.uiuc.edu

Abstract

The architecture of World Wide Web (WWW) browsers and servers support full file transfer for document retrieval. TCP is used for data transfers by Web browsers and their associated Hypertext Transfer Protocol (HTTP) servers. Full file transfer and TCP are unsuitable for continuous media, such as real time audio and video. In order for the WWW to support continuous media, we require the transmission of video and audio *on-demand* and in *real time*, as well as *new protocols* for real time data. We extend the architecture of the WWW to encompass the dynamic, real time information space of video and audio. Our WWW browser *Vosaic*, short for *Video Mosaic*, incorporates real time video and audio into standard hypertext pages and which are displayed in place. Video and audio transfers occur in real time; there is no file retrieval latency. The video and audio result in compelling Web pages. Real time video and audio data can be effectively served over the present day Internet with the proper transmission protocol. We have developed a real time protocol called VDP that we specialize for handling real time video over the WWW. VDP minimizes inter-frame jitter and dynamically adapts to the client CPU load and network congestion. Our WWW server dynamically changes transfer protocols, adapting to the request stream and the meta-information in requested documents. Experiments show a forty-four-fold increase in received video frame rate (0.2 frames per second (fps) to 9 fps) with the use of VDP in lieu of TCP, with a commensurate improvement in observed video quality. Our work enables a *video-enhanced Web*.

1 Introduction

Traditional information systems design for World Wide Web clients and servers has concentrated on document retrieval and the structuring of document-based information, for example, through hierarchical menu systems as is used in gopher, or links in hypertext as in the Hypertext Markup Language (HTML)[2]. The architecture of current information systems on the WWW has been driven by the *static* nature of document-based information. This is reflected in the use of the file transfer mode of document retrieval and the use of stream-based protocols, such as TCP[1]. Full file transfer and TCP are unsuitable for continuous media, and we discuss the reasons below. In order to incorporate video and audio into the WWW, we have extended the architecture of the WWW to enable a *video-enhanced Web*.

The easy-to-use, point-and-click user interfaces of WWW browsers, first popularized by NCSA Mosaic, have been the key to the widespread adoption of HTML and the World Wide Web by the entire Internet community. Although traditional WWW browsers perform commendably in the static information spaces of HTML documents, they are ill-suited for handling continuous media, such as real time audio and video. Research in our laboratory has resulted in *Vosaic*, short for *Video Mosaic*, a tool that extends the architecture of vanilla NCSA Mosaic to encompass the dynamic, real time information space of video and audio. *Vosaic* incorporates real time video and audio into standard Web pages and the video is displayed in place. Video and audio transfers occur in real time; there is thus no retrieval latency. The user accesses real time sessions with the familiar “follow-the-link” point and click method. Mosaic was chosen as the software platform for our work because it is a widely available tool for which the source code is available, and with which we have significant experience.

Vosaic is a vehicle for exploring the integration of video with hypertext documents, allowing one to embed video links in hypertext. In Vosaic, sessions on the Multicast Backbone (Mbone)[6] can be specified using a variant of the Universal Resource Locator (URL) syntax. For example, the URL

mbone://224.2.252.51:4739:127:nv

encodes a Mbone transmission with a *Time To Live* (TTL) factor of 127, a multicast address of 224.2.252.51, a port address of 4739, and an *nv* video transmission format[12]. While our original intent was for Vosaic to support the navigation of the Mbone's information space, we have extended Vosaic to include real time retrieval of data from arbitrary video servers. Vosaic now supports the streaming and display of real time video, video icons and audio within a WWW hypertext document display. The Vosaic client adapts to the received video rate by discarding frames that have missed their arrival deadline. Early frames are buffered, minimizing playback jitter. Periodic resynchronization adjusts the playback to accommodate network congestion. The result is real time playback of video data streams.

Present day *httpd* servers exclusively use the TCP protocol for transfers of all document types. Our experiments indicate that real time video and audio data can be effectively served over the present day Internet with the proper choice of transmission protocols. The server uses an augmented Real Time Protocol (RTP)[13] called VDP with built-in fault tolerance for video transmission. Section 6 describes VDP. Feedback within VDP from the client allows the server to control the video frame rate in response to client CPU load or network congestion. The server also dynamically changes transfer protocols, adapting to the request stream and the meta-information in requested documents. Our initial experiments show a forty-four-fold increase in the received video frame rate (0.2 frames per second (fps) to 9 fps) with VDP in lieu of TCP, with a commensurate improvement in observed video quality. We describe the implementation and results of our experiments below.

In section 2, we enumerate the reasons why the current WWW architecture is highly unsuitable for continuous media, and discuss how video and audio may be effectively incorporated into the current WWW in section 3. Then in section 4, we describe the architecture of our prototype WWW client Vosaic, and in section 5, we describe the extended HTTP server that we have constructed. Our specialized video datagram protocol VDP is discussed in section 6. Experimental results are presented in section 7. Related work is discussed in section 8 and we give our conclusions in section 9.

2 Why the current WWW is unsuitable for Continuous Media

2.1 Full file transfer as a retrieval paradigm

Multimedia browsers such as Mosaic are excellent vehicles for browsing information spaces on the Internet that are made up of *static* data sets. However, attempts at the inclusion of video and audio in the current generation of multimedia browsers are limited to pre-recorded and canned sequences that are *retrieved* as full files. While the file transfer paradigm is adequate in the arena of traditional information retrieval and navigation, it becomes cumbersome for real time data. The transfer times for video and audio files can be very large. Video and audio files now on the Web take minutes to hours to retrieve, thus severely limiting the inclusion of video and audio in current Web pages, because the latency required before playback begins can be unacceptably long. The file transfer method of browsing also assumes a fairly static and unchanging data set for which a single uni-directional transfer is adequate for browsing some piece of information. Real time sessions such as videoconferences, on the other hand, are not static. Sessions happen in real time and come and go over the course of minutes to days.

2.2 TCP-based transfers

The Hypertext Transfer Protocol (HTTP) is the transfer protocol used between WWW clients and servers for hypertext document service. The HTTP uses TCP as the primary protocol for reliable document transfer. TCP is unsuitable for real time audio and video for several reasons.

- TCP imposes its own flow control and windowing schemes on the data stream. These mechanisms effectively destroy the temporal relations shared between video frames and audio packets.
- Reliable message delivery is not required for video and audio. Video and audio streams tolerate frame losses. Losses are seldom fatal although detrimental to picture and sound quality. TCP

retransmission causes further jitter and skew internally between frames and externally between associated video and audio streams.

3 How to get video and audio in the WWW

The use of both full file transfer and TCP as a transfer protocol is clearly unsuitable for supporting video and audio. We concluded that to truly support video and audio in the WWW, one requires:

- the transmission of video and audio *on-demand*, and in *real time*, as well as
- *new protocols* for real time data.

On demand, real time video and audio solves the problem of playback latency. In Vosaic, the video or audio is streamed across the network from the server to the client in response to a client request for a Web page containing embedded videos. The client plays the incoming multimedia stream in real time as the data is received in real time.

However, the real time transfer of multimedia data streams introduces new problems of maintaining adequate playback quality in the face of network congestion and client load. As the WWW is based on the Internet, resource reservation to guarantee bandwidth, delay or jitter is not possible. The delivery of IP packets across the international Internet is typically best effort, and subject to network variability outside the control of any video server or client.

Our initial effort focuses on supporting real time video in the Web. Inter-frame jitter greatly affects video playback quality across the network.¹ High jitter typically causes the video playback to appear “jerky.” In addition, network congestion may cause frames delays or losses. Transient load at the client side may prevent it from handling the full frame rate of the video. We created a specialized real time transfer protocol for handling video across the Internet. Our experiments indicate that the protocol successfully handles real time Internet video by minimizing jitter and incorporating dynamic adaptation to the client CPU load and network congestion.

4 Vosaic

4.1 Vosaic’s Roots

Vosaic is derived from NCSA Mosaic. Mosaic concentrates on HTML documents. While all media types are treated as documents, each media type is handled differently. Text and inlined images are displayed in place. Other media types, such as video and audio files, or special file formats (eg., postscript) are handled externally by invoking other programs. In Mosaic, documents are not displayed until fully available (see section 2.1).² The Mosaic client keeps the retrieved document in temporary storage until all of the document has been fetched. The sequential relationship between transferring and processing of documents makes the browsing of large video/audio documents and real time video/audio sources problematic. Transferring such documents require long delay times and large client side storage space. This makes real time playback impossible.

There is one argument that the functions of Mosaic should be kept at a minimum while additional functionality is left up to specialized external viewers. Work along this line has led to the development of the Common Gateway Interface[18], a standard for harnessing external programs to view files according to their MIME[9, 8] types. One key reason favoring the success of Mosaic is its incorporation of inlined images with text, resulting in a semantically rich environment. Leaving the display of audio and video to external viewers causes a loss of semantic content in a hypertext document that includes these media types. Real time video and audio convey more information if directly incorporated into the display of a hypertext document. For example, we have implemented real time *video menus* and *video icons* as an extension of HTML in Vosaic. Video menus present the user with several choices. Each choice is in the form of a moving video. One may, for example, click on a video menu item to follow the link, and watch the clip in full size. Video icons show a video in an small, unobtrusive icon-sized rectangle within the HTML document. Our experience indicates that embedded real time video within WWW documents greatly enhances the look and feel of a Vosaic page. Video menu items convey more information about the choices available than simple textual descriptions or static images.

¹For our purposes, we take jitter as the variance in inter-arrival time between subsequent frames of a video stream.

²More modern browsers, such as Netscape[10], incrementally display a document as it is retrieved.

4.2 Vosaic's Internal Structure

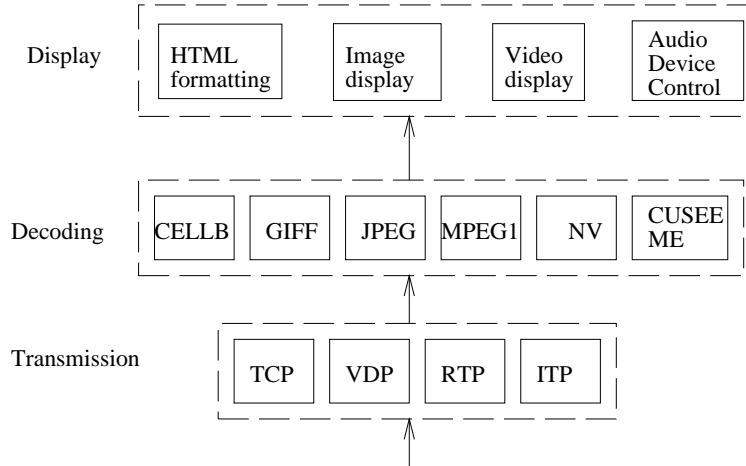


Figure 1: Vosaic's internal structure.

HTML documents with video and audio integrated are characterized by a variety of data transmission protocols, data decoding formats, and device control mechanisms (eg., graphical display, audio device control, and video board control). Vosaic has a layered structure to meet these requirements. The layers are depicted in figure 1. They are:

- document transmission,
- document decoding, and
- document display.

A document data stream flows through these three layers by using different components from different layers. The composition of components along the data path of a retrieved document occurs at run-time according to document meta-information returned by our extended HTTP server.

As discussed in the previous section, TCP is only suitable for static document transfers, such as text and image transfers. Real time playback of video and audio requires other protocols. The current implementation in the Vosaic document transmission layer includes TCP, VDP and RTP. Vosaic is configured to have TCP support for text and image transmission. Real time playback of real time video and audio uses VDP. RTP is the protocol used by most Mbone conferencing transmissions. A fourth protocol under consideration is for interactive communication (used for virtual reality, video games and interactive distance learning) between the web client and server.

The decoding formats currently implemented include:

- For images:
 - GIF
 - JPEG
- For video:
 - MPEG1
 - NV
 - CUSEEME
 - Sun CELLB
- For audio:
 - AIFF
 - MPEG1

MPEG1 includes support for audio embedded in the video stream,

The display layer includes traditional HTML formatting and inline image display. We have extended the display to incorporate real time video display and audio device control.

4.3 HTML and URL Extensions

Standard URL specifications include FTP, HTTP[3], WAIS[17], and others, covering most of the currently existing document retrieval protocols. However, access protocols for video and audio conferences on the Mbone is not defined and not supported. We have extended the standard URL[4] specification and HTML to accommodate real time continuous media transmission. The extended URL specification supports Mbone transmission protocols using the **mbone** keyword as a URL scheme and on-demand continuous media protocols using **cm** as the URL scheme. The format of the URL specifications for the Mbone and continuous real time are as follows:

```
mbone://address:port:ttl:format
cm://address:port:format/filepath
```

Examples are given below:

```
mbone://224.2.252.51:4739:127:nv
cm://showtime.ncsa.uiuc.edu:8080:mpegvideo/puffer.mpg
cm://showtime.ncsa.uiuc.edu:8080:mpegaudio/puffer.mp2
```

The first URL encodes an Mbone transmission on the address 224.2.252.51, on port 4739, with a time to live factor of 127, using *nv* format video. The second and third URLs encode continuous media transmissions of MPEG video and audio respectively.

Incorporating inline video and audio in HTML necessitates the addition of two more constructs to the HTML syntax. The additions follow the syntax of inline images closely. Inlined video and audio segments are specified as follows:

```
<video src="address:port/filepath" option="control—cyclic">
<audio src="address:port/filepath" option="control—cyclic">
```

The syntax for both video and audio is made up of a *src* part and an *options* part.

- *Src* specifies the server information including the address and port number.
- *Options* specifies how the media is to be displayed. Two options are possible: *control* or *cyclic*.
 - *Control* pops up a window with a control panel and the first frame of the video is displayed, with further playback controlled by the user.
 - *Cyclic* displays the video or audio clip in a loop. The video stream may be cached in local storage to avoid further network traffic after the first round of display. This is feasible when the size of video or audio clip is small. If the segment is too large to be stored locally at the client end, the client may also request the source to repeatedly send the clip. Cyclic video clips are useful for constructing *video menus* and *video icons*.

4.4 Control Interface

If the *control* keyword is given, a control panel is presented to the user. The control interface allows users to browse and control video clips. We provide following user control buttons.

1. *Play*: Start to play the video.
2. *Replay*: Start to play the video from the very beginning.
3. *Pause*: Pause the playing of the video.
4. *Stop*: Ends the playing of the video.
5. *Fast Forward*: Play the video at a faster speed. We implement this by dropping frames at the server site.

4.5 Video Widgets and Client/Server Interaction

A new video widget is required in order to display the video. Video widget creation follows three steps.

1. Before widget creation, the client communicates with the server asking for the size information of the video stream.
2. On receipt of the size information, the browser creates the video widget.
3. After the widget creation, the browser forks a video player process. The player process receives and displays the video within the video widget.

Real time video and audio use VDP as a transfer protocol. Control information exchange uses a TCP connection between the client and server.

5 Server

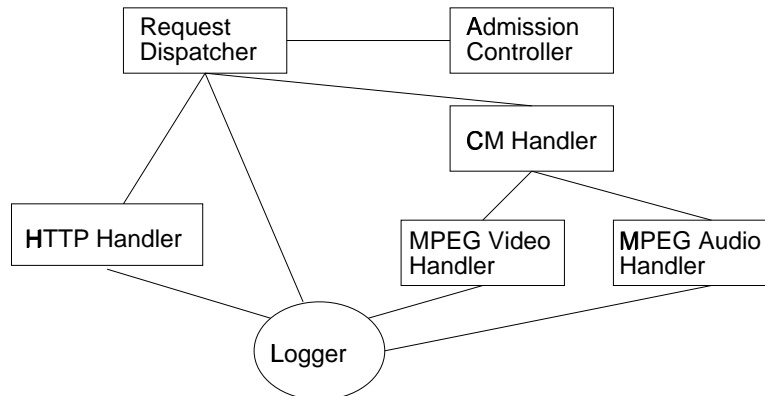


Figure 2: Server structure.

Vosaic works in conjunction with an extended HTTP server. The server uses the same set of transmission protocols as does Vosaic. It spawns a child to handle the transmission of each video stream. Video and audio are transmitted with VDP. Frames are transmitted at the originally recorded frame rate of the video. The server uses a feed forward and feedback scheme to detect network congestion and automatically delete frames from the stream in response to congestion.

The main components of the server are shown in figure 2. They are:

1. the main request dispatcher,
2. the admission controller,
3. the HTTP request handler,
4. the continuous media handlers, and
5. the server logger.

5.1 Admission Control

The *Main request dispatcher* receives client requests. The main request dispatcher passes the request to the *admission controller*. The admission controller then determines or estimates the requirements of the current request, which may include network bandwidth and CPU load. It then makes a decision on whether the current request should be served based on its knowledge of current conditions.

Traditional HTTP servers can do without admission control because document sizes are small, and request streams are bursty[16]. Requests are simply queued before service, and most documents can be handled quickly. In contrast, with continuous media transmissions in a video server, file sizes are large, and real time data streams have stringent time constraints. The server must ensure that it has enough network bandwidth and processing power to maintain service qualities to current requests. The criteria used to evaluate requests may be based on the requested bandwidth, server available bandwidth, and system CPU load. Our current system simply limits the number of concurrent streams to a fixed number. However, the admission control policy is flexible and can be made more sophisticated.

5.2 Request Handling

Once the system grants the current request, the main request dispatcher hands the request to one of several specific *request handlers*. The server currently has handlers for:

- HTTP,
- MPEG Video, and
- MPEG Audio.

Each handler uses a different transmission protocol for handling requests. HTTP requests use TCP/IP. MPEG video and audio use VDP (described in section 6). We have striven to ensure that the server design is flexible enough to incorporate more protocols.

5.3 Logging

The *logger* is responsible for recording the request and transmission statistics. Reed et al[16], McGrath[11] and Mogul[7] have studied the access patterns of the current Web servers. We expect that the access patterns for a video enhanced Web server will be substantially different from that of traditional WWW servers that supporting mainly text and static images. The statistics for the transmission of continuous media is recorded by the server logger in order that we can better understand the behavior of requests for continuous media. The statistics include the network usage and processor usage of each request, the quality of service data such as frame rate, frame drop rate, and jitter. The data will guide the design of future busy Internet video servers. These statistics are also important for analyzing the impact of continuous media on operating systems and the network.

6 VDP

VDP is an augmented real time datagram protocol we developed to handle video and audio over the WWW. The design of VDP is based on making efficient use of the available network bandwidth and CPU capacity for video processing. It is different from RTP in that it takes advantage of the point-to-point connection between Web server and Web client. The server end of VDP receives feedback from the client and adapts to the network condition between client and server and the client CPU load. VDP uses an *adaptation algorithm* to find the optimal transfer bandwidth. A *demand resend algorithm* handles frame losses.

6.1 VDP Channels

VDP is an asymmetric protocol. A VDP connection has two distinguished endpoints belonging to a client and a video server. There are two channels associated with each VDP connection. The first is an unreliable data transmission channel used for transmitting data and uncritical feedback information. Another one is a reliable channel used to send control information from the client to the server. Control information includes playback control, such as *play*, *stop*, *fast forward*, *rewind*, as well as connection management control, such as *connection termination*.

6.2 Transmission Mechanism

After the admission controller grants the request from the client, the server waits for the *play* command from the client. Upon receiving the *play* command, the server starts to send the video frames on the data channel using the *recorded frame rate*. The server end breaks large frames into smaller packets (currently of size 8 kilobytes), and the client end reassembles the packets into frames. Each frame is time-stamped by the server and buffered at the client side. The client controls the sending of frames by sending server control commands, like *stop* or *fast forward*, on the control channel.

6.3 Adaptation Algorithm

The adaptation algorithm dynamically adapts the video transmission rate to network conditions along the network span from the client to the server, as well as to the client end's processing capacity. The algorithm degrades or upgrades the server transmission rate depending on feed forward and feedback messages exchanged on the control channel. Protocols for the transmission of continuous media over the Internet must preserve network bandwidth as much as possible. If a client does not have enough processor capacity, it may not be fast enough to decode video and audio data. Network connections may also impose constraints on the frame rate at which video data can be sent. In such cases, the server must gracefully degrade the quality of service. The server learns of the status of the connection from client feedback. Feedback messages are of two types:

- **Frame drop rate**

The frame drop rate corresponds to frames received by the client but dropped because the client did not have enough CPU power to keep up with decoding the frames.

- **Packet drop rate**

The packet drop rate corresponds to frames lost in the network due to network congestion.

If the client side protocol discovers that the client application is not reading received frames quickly enough, it updates the frame loss rate. If the loss rate is severe, it sends the information to the server. The server then adjusts its transmission speed accordingly. In our current implementation, the system degrades if the loss rate exceeds 15%, and upgrades if the loss rate is below 5%.

In response to a video request, the server begins by sending out frames using the recorded frame rate. The server inserts a special packet in the data stream indicating the number of packets sent out so far. On receiving the feed forward message from the server, the client may then calculate the packet drop rate. The client returns the feedback message to the server on the control channel. In our implementation, feedback occurs every 30 frames. Our experiments indicate that adaptation occurs very quickly in practice — on the order of a few seconds.

6.4 Demand Resend Algorithm

The compression algorithms in some media formats use inter-frame dependent encoding. For example, a sequence of MPEG video frames has *I*, *P*, and *B* frames.[5]

- *I* frames are intra-frame coded with JPEG compression.
- *P* frames are predictive coded with respect to a past picture.
- *B* frames are bidirectionally predictive coded.

MPEG frames are arranged into groups with sequences that correspond to the pattern *I B B P B B P B B*. The *I* frame is needed by all *P* and *B* frames in order to be decoded. The *P* frames are needed by all *B* frames. This encoding method makes some frames more important than the others. The display quality is strongly dependent on the receipt of important frames. Since data transmission is unreliable over the Internet, there is a possibility of frame loss. If, in a sequence group of MPEG video frames *I B B P B B P B B* recorded at 9 frames/sec, the *I* frame is lost, the entire sequence becomes undecodable. This produces a one second gap in the video stream. In VDP, the responsibility of determining which frames are resent is put on the client based on its knowledge of the encoding format used by the video stream. In an MPEG stream, it may thus choose to request retransmissions of only the *I* frames, or of both the *I* and *P* frames, or all frames. VDP employs a buffer queue at least as large as the number of frames required during one round trip time between the client and the server. The buffer is full before the protocol begins handing frames to the client from the queue head. New frames enter at the queue tail. A demand resend algorithm is used to generate resend requests to the server in the event a frame is missing from the queue tail. Since the buffer queue is large enough, it is highly likely that resent frames can be correctly inserted into the queue before the application requires it.

7 Experimental Results

We carried out several experiments over the Internet. Our test data set consisted of four MPEG movies, digitized at rates ranging from 5 to 9 fps, with pixel resolution ranging from 160 by 120 to 320 by 240. Table 1 tabulates the test videos. The videos ranged from a short 14 second segment to one of several

Name	Frame Rate (fps)	Resolution	Number Of Frames	Play Time (secs)
model.mpg	9	160 by 120	127	14
startrek.mpg	5	208 by 156	642	128
puffer.mpg	5	320 by 240	175	35
smalllogo.mpg	5	320 by 240	1622	324

Table 1: MPEG test movies.

minutes duration.

In order to observe the playback video quality, we based the client side of all our tests in our laboratory in Urbana, Illinois, USA. In order to cover the widest possible range of configurations, we set up servers corresponding to local, regional and international sites relative to our geographical location. We used a server at the National Center for Supercomputing Applications (NCSA) for the local case. NCSA is connected to the local campus network via Ethernet. For the regional case, we used a server at the University of Washington on the east coast of North America, in Washington State. Finally, a copy of

Name	IP Address	Function
indy1.cs.uiuc.edu	128.174.240.90	local client
showtime.ncsa.uiuc.edu	141.142.3.37	local server
agni.wtc.washington.edu	128.95.73.229	regional server
gloin.ifi.uio.no	129.240.106.13	international server

Table 2: Hosts used in our tests.

Name	% Dropped Frames	Jitter (ms)
model	0	8.5
startrek	0	5.9
puffer	7.5	43.6
smallogo	0.5	22.5

Table 3: Local test.

our server was set up at the University of Oslo in Norway to cover the international case. Table 2 lists the names and IP addresses of the hosts used for our experiments.

Tables 3, 4, and 5 show the results for sample runs using the test videos by our Web client accessing the local, regional and international servers respectively. Each test involved the Web client retrieving a single MPEG video clip. We used an unloaded SGI Indy as the client workstation. The numbers give the average frame drop percentage and average application-level inter-frame jitter in milliseconds for thirty test runs. Frame rate changes due to the adaptive algorithm was seen in only one run. That run used the **puffer.mpg** test video in the international configuration (Oslo, Norway to Urbana, USA). The frame rate dropped from 5 fps to 4 fps at frame number 100, then increased from 4 fps to 5 fps at frame number 126. The rate change indicated that transient network congestion caused the video to degrade for a 5.2 second period during the transmission.

The results indicate that the Internet can in fact support a video-enhanced Web service. Inter-frame jitter in the local configuration is negligible, and below the threshold of human observability (usually 100 ms) in the regional case. Except for the **puffer.mpg** runs, the same holds true for the international configuration. In that case, the adaptive algorithm was invoked because of dropped frames and the video quality was degraded for a 5.2 second interval. The VDP buffer queue efficiently minimizes frame jitter at the application level.

Our last test exercised the adaptive algorithm more strongly. We used the local configuration and retrieved a version of **smallogo.mpg** recorded at 30 fps at a pixel resolution of 320 by 240. This is a high quality video clip at a medium size, requiring significant computing resources for playback. Figure 3 shows a graph of frame rate versus frame sequence number for the server transmitting the video. The client side buffer queue was set at 200 frames, corresponding to about 6.67 seconds of video. The buffer at the client side first fills up, and the first frame is handed to the application at frame number 200. The client workstation does not have enough processing capacity to decode the video stream at the full 30 fps rate. The client side protocol detects a frame loss rate severe enough to report to the server at frame number 230. Our current criteria for degrading the transmission is when the frame loss rate exceeds 15%. The transmission is upgraded if the loss rate is below 5%.

The server begins degrading its transmission at frame number 268, that is, within 1.3 seconds from the client detecting that its CPU was unable to keep up. The optimal transmission level was reached in 7.8 seconds, corresponding to a 9 frame per second transmission rate. Stability was reached in a further

Name	% Dropped Frames	Jitter (ms)
model	0	46.3
startrek	0	57.1
puffer	0	34.3
smallogo	0.2	50.0

Table 4: Regional test.

Name	% Dropped Frames	Jitter (ms)
model	0	20.1
startrek	0	22.0
puffer	19	121.4
smallogo	0.8	46.7

Table 5: International test.

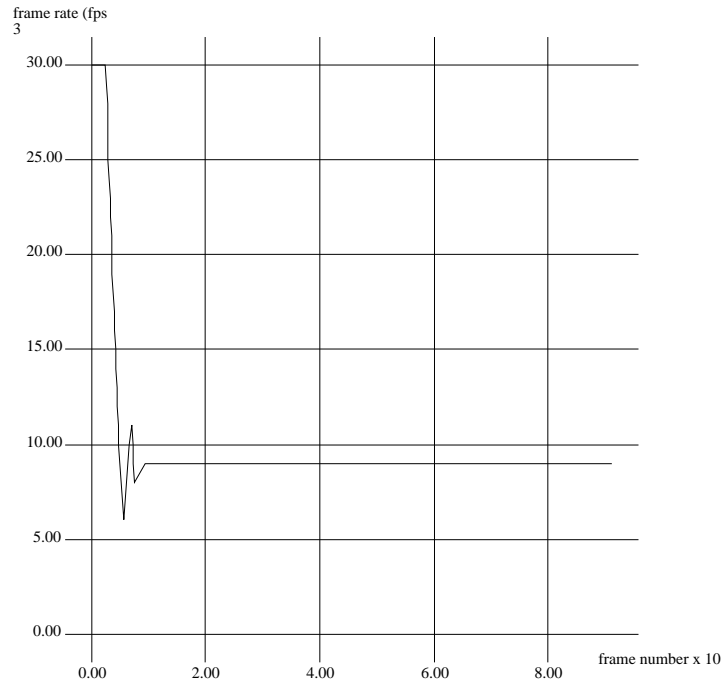


Figure 3: Frame rate adaptation for smallogo.mpg.

14.8 seconds. The deviation from optimal did not exceed 3 frames per second in either direction during that period. The results show a fundamental tension between large buffer queue sizes that minimize jitter and server response times.

The test with very high quality video at 30 fps with a frame size of 320 by 240 represents a pathological case. However, the results show that the adaptive algorithm is an attractive way to reach optimal frame transmission rates for video in the WWW. The test implementation changes the video quality by 1 frame per second at each iteration. We are experimenting with non-linear schemes based on more sophisticated policies.

8 Related Work

A major impediment to browsing Web documents which include video and audio clips is the long transfer latency for continuous media files. The commercial successors to Mosaic like Netscape[10] attempt to minimize the transfer latency for large documents by incrementally displaying the document as it is retrieved. Unfortunately, this idea is not carried through to continuous media.

Sun Microsystem's HotJava[15] product introduced a novel method for the inclusion of animated multimedia in a Web browser. HotJava allows the browser to download executable scripts written in the Java programming language. The execution of the script at the client end enables the animation of graphic widgets within a Web page. In contrast, we have concentrated on the streaming of real time video and audio over the WWW.

Smith's CM Player[14] is a networked continuous media toolkit. It is used as part of an experimental

video-on-demand (VOD) system at UC Berkeley. CM Player is a stand alone system. In contrast, we have cast our work in a WWW context, incorporating continuous media into Web browsers and servers. CM Player employs Cyclic-UDP for the transport of video streams between the VOD server and the CM Player client. Frames are prioritized at the server, and clients request resends on detecting frame losses. Cyclic-UDP repeatedly resends high priority frames in order to give them a better chance of reaching the destination. VDP's demand resend algorithm is similar to Cyclic-UDP, except that the client decides which frames get retransmitted. In an MPEG transmission, the client can decide to tolerate the loss of B frames but require the resend of all I frames.

9 Conclusion

In order to integrate real time continuous media into the WWW, we extended the traditional full file transfer paradigm of the WWW to include real time transfer and browsing of video- and audio-enhanced HTML documents. The motivation behind Vosaic as a research tool is its integration of real time multimedia information into an easy-to-use WWW client. The new video datagram protocol VDP is designed for video data transfer over the Internet. Our experiments conclude that it is possible to transmit high quality video over the present day Internet with VDP.

Our work indicates that a *video-enhanced* World Wide Web is indeed possible.

10 Acknowledgements

David Putzolu, together with See-Mong Tan, was one of the originators of the Vosaic idea. We thank Bruce Schatz for sparking our interest in information systems design. We are grateful to our partners at NCSA for supporting our work on a video-enhanced Web. In particular, we thank Charlie Catlett and Jeff Terstriep at NCSA. Ellard Roush provided helpful comments at every stage. We thank our friends Jisheng Liang at the University of Washington and Dong Xie at the University of Oslo for their generous help with our experiments.

References

- [1] D. Comer and D. Stevens. *Internetworking with TCP/IP*. Prentice Hall, Englewood Cliffs, NJ, 1991.
- [2] World Wide Web Consortium. Hypertext Markup Language. Available on the WWW via <http://www.w3.org/hypertext/WWW/MarkUp>.
- [3] World Wide Web Consortium. Hypertext Transfer Protocol. Available on the WWW via <http://www.w3.org/hypertext/WWW/Protocols>.
- [4] World Wide Web Consortium. Uniform Resource Locators. Available on the WWW via <http://www.w3.org/hypertext/Addressing/URL>.
- [5] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, April 1991.
- [6] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANS. *ACM Transactions on Computer Systems*, pages 85–110, May 1990.
- [7] J.C. Mogul. Operating Systems Support for Busy Internet Services. In *Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, WA, May 1995. IEEE Computer Society.
- [8] K. Moore. MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text. Internet RFC 1522, September 1993.
- [9] N. Borenstein and N. Freed. MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. Internet RFC 1521, September 1993.
- [10] Netscape Communications, Inc. Netscape. Available on the WWW via <http://home.netscape.com>.
- [11] R. E. McGrath. What We Do and Don't Know About the Load on the NCSA WWW Server. Available on the WWW via <http://www.ncsa.uiuc.edu/InformationServers/Colloquia/28.Sep.94/Begin.html>, September 1994.

- [12] Ron Frederick. Experiences with software real time video compression. Technical report, Xerox Palo Alto Research Center, July 1992. Available on the WWW via <ftp://parcftp.xerox.com/pub/net-research/nv-paper.ps>.
- [13] H. Schulzrinne and S. Casner. RTP: A Transport Protocol for Real time Applications. Internet Draft, October 1993.
- [14] Brian Smith. *Implementation Techniques for Continuous Media System and Applications*. PhD thesis, University of California, Berkeley, 1993.
- [15] Sun Microsystems, Inc. Hot Java. Available on the WWW via <http://www.sun.com>.
- [16] Thomas T. Kwan and Robert E. McGrath and Daniel A. Reed. User Access Patterns to NCSA's World Wide Web Server. Technical report, University of Illinois at Urbana-Champaign, 1995. Available on the WWW via <http://www-pablo.cs.uiuc.edu/Projects/Mosaic/mosaic.html>.
- [17] Trans-European Research and Education Networking Association. WAIS. Available on the WWW via <http://www.earn.net/gnrt/wais.html>.
- [18] World Wide Web Consortium. Common Gateway Interface. Available on the WWW via <http://www.w3.org/hypertext/WWW/Overview.html>.