

## QoS management in a World Wide Web environment which supports continuous media

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

1997 Distrib. Syst. Engng. 4 38

(<http://iopscience.iop.org/0967-1846/4/1/005>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 38.107.179.212

The article was downloaded on 20/02/2012 at 21:04

Please note that [terms and conditions apply](#).

# QoS management in a World Wide Web environment which supports continuous media

Michael Fry<sup>†¶</sup>, Aruna Seneviratne<sup>‡+</sup>, Andreas Vogel<sup>§\*</sup> and Varuni Witana<sup>||#</sup>

<sup>†</sup> School of Computing Sciences, <sup>‡</sup> School of Electrical Engineering and <sup>||</sup> CRC for Distributed Systems Technology, University of Technology, Sydney, Sydney 2007 Australia

<sup>§</sup> CRC for Distributed Systems Technology, University of Queensland, Brisbane 4072, Australia

**Abstract.** Until recently the World Wide Web (WWW) and associated browser technologies have provided no support for the delivery of continuous media in real time. Instead, files have been down-loaded completely and then played back. There are now some systems available that provide support for streaming of audio and/or video between WWW servers and clients. However the Internet comprises a vast range of networks and end systems with different characteristics, which together determine the level of continuous media service quality that can be supported for a given connection. A quality of service (QoS) model is required that will encompass the heterogeneity of the Internet environment.

In this paper we present our approach to delivering continuous media in WWW environments, based on a model of end-to-end QoS management. We have developed an experimental implementation that permits real-time delivery of audio and video within the WWW. Significantly our implementation incorporates end-to-end QoS negotiation and QoS adaptation, which adjusts the delivery stream to match prevailing network and system conditions. We describe the design and implementation of our WWW-based QoS model. We then discuss some issues arising from our experience, including a proposal for future work based on a more open, object oriented architecture.

## 1. Introduction

### 1.1. Motivation

The World Wide Web (WWW) and associated browser technologies currently have little support for the delivery of continuous media (e.g. video and audio) in real time. Instead, media files are usually first down-loaded in their entirety, before being played out at the client by an external viewer. For anything other than trivially small video clips this is an unacceptably lengthy process.

The Web has experienced phenomenal growth in the last few years. WWW browsers such as Netscape and Mosaic are becoming ubiquitous, user-friendly interfaces to the Internet.

Therefore, enhancement of WWW technologies to support real-time, continuous media delivery seems attractive. Furthermore the power of networking and workstation hardware is now such that the Internet can support multimedia conferencing on a daily basis using

tools such as *vic* and *vat* [9,3], albeit at varying levels of quality.

There have been systems developed recently to support audio and/or video streaming. Examples include RealAudio and StreamWorks. However there are some difficulties facing the delivery of continuous media that are inherent to the WWW and its underlying Internet transport system. The 'stateless' HTTP model is not well suited to the delivery of continuous media. More fundamentally, the Internet is a highly heterogeneous, best-effort network. Furthermore the client and server end-systems have highly heterogeneous characteristics that affect the level of information delivery that can be supported across arbitrary connections. However the Internet has evolved to a point where transactions can be made deterministic and negotiable. Developments are occurring through the deployment of ATM networks and/or resource-reservation oriented protocols such as ST-II and RSVP [11, 22].

The notion of *end-to-end quality of service (QoS)* [21] has emerged recently to manage network diversity and the heterogeneity of end-systems, while attempting to deliver an optimal level of continuous media. Broadly, under this notion end-to-end QoS is a reflection of user perceptions. QoS is thus determined by a number of factors including the

¶ E-mail address: mike@socs.uts.edu.au

+ E-mail address: aps@ee.uts.edu.au

\* E-mail address: andreas@dstc.edu.au

# E-mail address: varuni@dstc.uts.edu.au

capabilities of the end-systems, the load on those systems, the bandwidth of the Internet connection, and the user's own preferences for information delivery.

A number of models of end-to-end QoS management have been developed, for example [2, 15, 7]. Each has a fundamental notion of scalable and adaptable delivery to match available resources. However, to our knowledge there has been little experimental implementation of such models. Thus the primary motivation for our work has been to test the use of end-to-end QoS management in a real environment. Because of its pervasiveness, the environment we have chosen is the World Wide Web.

## 1.2. Objectives and contents

This motivation has driven an applied research project. The objectives of the first stage of this project (reported in this paper) were as follows.

- To investigate the use of quality of service management in the WWW.
- To define an architecture which enables real-time delivery of continuous media in the WWW.
- To specify a negotiation protocol and an adaptation scheme that will facilitate end-to-end QoS management.
- To implement a proof-of-concept case-study.

The rest of this paper is organized as follows. In the next section we describe our initial architecture for integrating QoS negotiation and continuous media delivery within the WWW context. We then discuss in detail our design and implementation of end-to-end QoS management and the stream protocol. Our approach to QoS has been discussed elsewhere [15]. Here we focus on the details of implementation. User interface issues are discussed. We then discuss and evaluate some of our implementation experiences. This highlights future work to be done. We finish by considering related work and the status of the project.

## 2. Architecture

The World Wide Web is founded on the HyperText Transfer Protocol (HTTP) which determines the rules of communication between WWW servers and clients. HTTP uses TCP as its underlying transport protocol, which in turn uses IP at the internetwork layer. TCP is used for all communication between the client (WWW browser) and server, including content selection and content delivery (text and images). Thus all HTTP communication displays the reliability semantics of TCP. The WWW also permits the invocation of other protocols such as file transfer (FTP) and mail (SMTP). These protocols also use TCP.

Thus while there are other transport protocols deployed on the Internet like UDP and RTP (see section 5), these are not utilized within the current WWW architecture, which is solidly based on TCP.

We identified three types of 'protocol' that we would need for our implementation:

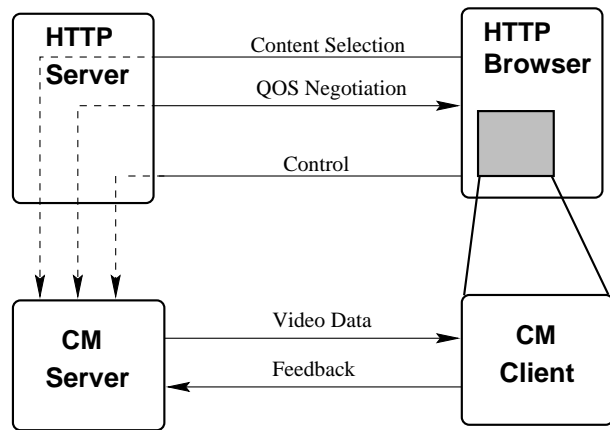


Figure 1. Initial architecture.

- HTTP, for communication between WWW server and client, and possibly including video content selection and user control over delivery (start/stop)
- a 'stream' protocol for continuous media delivery, and associated feedback signals for QoS adaptation
- a QoS negotiation protocol.

The current set of WWW protocols are not suitable for real-time delivery of continuous media. This is primarily due to the well known unsuitability of the underlying transport protocol TCP. Thus HTTP/TCP could not be used for our 'stream' connection.

Likewise, the 'stateless' HTTP paradigm is not well suited to sequential interactions such as those required to incrementally negotiate and manage QoS between a client and a server. Thus to fully integrate QoS management and real-time delivery within the WWW model we were faced with having to modify the HTTP interactions, with consequent modifications to both client and server. However this would have led to lost conformance and interoperability. As a design philosophy we wished as far as possible to maintain interoperability with existing WWW clients and servers. Thus we have initially extended the HTTP client-server model as illustrated in figure 1.

There are two new components, a continuous media (CM) server and a corresponding client. The CM server is controlled from the WWW user (HTTP client) via the HTTP server using the Common Gateway Interface (CGI). This applies also to the QoS negotiation. Thus for our first stage all the QoS negotiations are visible to and controlled by the user via forms (this is discussed further below).

There is an additional connection between the CM server and the CM client. Over this connection the continuous media data are delivered and the feedback for QoS adaptation is returned.

## 3. The QoS management model

The Internet, as the WWW's underlying infrastructure, has an impact on the QoS levels attainable by WWW applications in two ways. Firstly, bandwidth can range

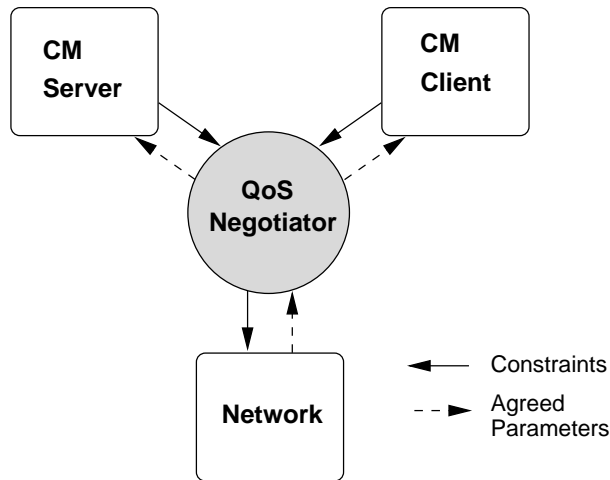


Figure 2. QoS negotiation—objective.

from low-speed modem connections (kilobit/s) to high-speed networks (gigabit/s). Secondly, the TCP/IP protocol family shares available resources evenly amongst all competing connections.

An additional impact on the overall QoS level of WWW applications comes from the characteristics of the end-systems, both the server and the client. Examples of such characteristics on the server side are the delay and jitter caused by access to storage devices or process scheduling. On the client side parameters of the monitor (colour depth) or the availability of additional hardware affect the QoS level attainable.

Finally, the format of the delivered data has an additional impact on the level of QoS. Examples are postscript versus HTML for text, or H.261 versus MPEG for video.

### 3.1. QoS negotiation

The QoS negotiation protocol aims for the best compromise between capabilities and constraints of the CM server, the network and the CM client, and particular user requirements and preferences. As a result some initial parameter values (QoS parameters) and a degradation path are calculated.

The degradation path is an ordered list where each element describes a threshold and degraded parameters (described further below). This list is determined by the user's preferences, e.g. keep image size over frame-rate (for video).

For adaptation, the CM client monitors the actual parameter values. When they fall below a certain threshold as specified in the degradation path, the QoS level is degraded in accordance with QoS parameters.

Before the video can be delivered negotiation needs to take place to determine common QoS parameters and parameter values. The parties to this negotiation process are shown in figure 2. These parties are the the CM server the CM client and the network. Each party in this negotiation presents their constraints. At the end of the negotiation an agreed set of QoS parameter values is determined, as described below.

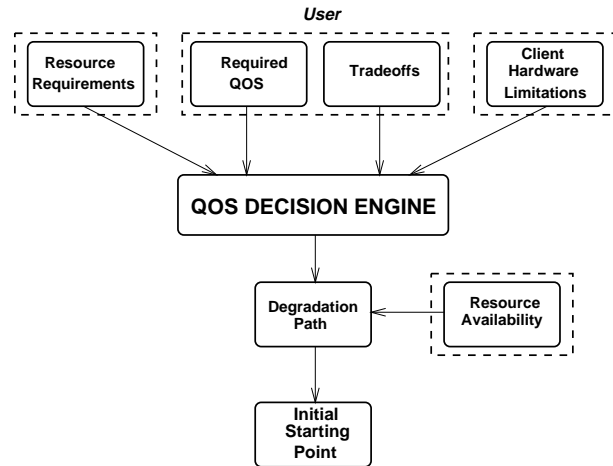


Figure 3. QoS decision engine.

### 3.2. QoS levels

The QoS levels supported for the video correspond to user perceptible parameters such as frame rate, image size, colour and image quality. Currently frame rates of 25, 20, 15, 10 and 5 frames/s and two sizes: large ( $320 \times 240$ ) and small ( $160 \times 120$ ) are supported. Different image quality levels are not currently available but it is possible to support this, e.g. for H.261 by adjusting the quantization levels [1]. Some user perception tests need to be performed to evaluate suitable levels of image quality. Black and white video is also not currently supported.

In addition to the various supportable QoS levels, it is possible to use different encoding schemes within these QoS levels. The encoders currently supported are JPEG and H.261. These were chosen primarily because of availability—they are supported by the *vic* video player used for our implementation [9]. The JPEG format encodes each frame as a self-contained image (sometimes called moving JPEG). H.261 uses a similar (DCT-based) coding method for frames, but also employs inter-frame coding for sequential frames with high temporal redundancy. That is, it can encode the difference between successive frames.

We were keen to explore the characteristics of different encoding methods and their impact on QoS. The same QoS level for the two encodings may have different resource requirements. For example H.261 typically achieves a better compression ratio, and therefore has a lower bandwidth requirement than JPEG at the same QoS level. However for higher frame rates we require hardware decoding, which is currently only widely available for JPEG.

## 4. The QoS decision engine

The QoS decision engine is the heart of the QoS negotiation process. Its inputs and outputs are shown in figure 3. These are explained in more detail in the following sections.

**Table 1.** Bandwidth requirements for available QoS levels and encodings in  $\text{Kb s}^{-1}$ .

Frame rate	VJ day				Organ transplant			
	JPEG		H.261		JPEG		H.261	
	Large	Small	Large	Small	Large	Small	Large	Small
25	2560	832	—	—	1741	533	—	—
20	2048	646	—	—	1434	449	—	—
15	1536	488	—	—	1024	331	—	—
10	1024	330	504	304	681	220	297	141
5	504	162	442	156	344	110	220	82

**Table 2.** Hardware limitations.

Video encoder	Maximum frame rate (frames/s)				
	Alpha JPEG H/W	Alpha 250 4/266	Alpha 5000/400	DEC 5000/240	DEC 5000/25
JPEG large	25	20	10	—	—
JPEG small	25	25	25	15	5
H.261 large	N/A	10	10	5	—
H.261 small	N/A	10	10	10	10

#### 4.1. Inputs

- *Resource requirements* The amounts of resources needed to maintain a particular QoS level are inputs to the negotiation. The only resource currently considered is network bandwidth. The necessary bandwidth is provided by the compiler of the video clip. They should be given for the different QoS levels and encoding options available. In the prototype the peak bandwidth requirement is specified. The bandwidth requirements for two video clips are shown in table 1. It can be seen that the bandwidth requirements are clip dependent. The first clip (VJ Day) contains many crowd scenes that seem to reduce spatial redundancy. It also contains more movement (temporal) than the other clip.
- *Required QoS level* The user may specify his/her desired QoS level in terms of frame rate, size etc. Factors such as cost might mean a user would not always ask for the maximum QoS level.
- *Tradeoffs* The user specifies his/her QoS preferences to weight the above QoS parameters. When the desired QoS level cannot be met the least important QoS parameter is sacrificed. For example a user could specify that maintaining a high frame rate is more important than maintaining size.
- *Client hardware limitations* This would limit the video options that can be received on a particular hardware configuration. Table 2 shows the hardware capabilities of the machines available in our laboratory. As can be seen from the table, a frame rate of 25 frames/s can only be supported if a hardware JPEG decoder is available. If software decoding is used the maximum frame rate achievable is dependent on the encoding type and the size of the video window. This information has been derived from our measurements.

**Table 3.** Degradation path—size over frame rate.

Format	Size	Frame rate	Bandwidth ( $\text{Kb s}^{-1}$ )
JPEG	large	25	2560
JPEG	large	20	2048
JPEG	large	15	1536
JPEG	large	10	1024
H.261	large	10	504
H.261	large	5	442
JPEG	small	10	330
H.261	small	10	304
JPEG	small	5	163
H.261	small	5	156

**Table 4.** Degradation path—frame rate over size.

Format	Size	Frame rate	Bandwidth ( $\text{Kb s}^{-2}$ )
JPEG	large	25	2560
JPEG	small	25	832
JPEG	small	20	646
JPEG	small	15	488
JPEG	small	10	330
H.261	small	10	304
JPEG	small	5	163
H.261	small	5	156

#### 4.2. Outputs

- *Degradation path* When the inputs shown above are entered into the QoS decision engine the result is a degradation path. This is an ordered list of available QoS levels and encodings and their resource requirements, in this case bandwidth. The list is limited by what the user desires and by hardware constraints.

The degradation path varies according to what the user's tradeoffs are. Degradation paths corresponding to two different tradeoffs are shown in tables 3 and 4. In table 3 the user has specified that size is more important than frame rate. As can be seen from the table, as the available bandwidth decreases it is necessary to drop the frame rate in order to maintain the large size. In table 4 the user has specified that frame rate is more important than size. Hence the video will drop from 25 frames/s large to 25 frames/s small in order to maintain the user preference.

- *Initial starting point* Once the degradation path is determined a suitable starting point in the path is selected. This depends on current resource availability.

For the first prototype the only resource we have considered is network bandwidth. We obtain an approximate estimate of the available bandwidth between the server and the client by using the Unix *ping* command, which returns a round-trip time measurement for the transmission of a specified amount of data. (This is a crude solution, used for prototyping purposes.) This is then used to determine the starting point in the degradation path.

#### 4.3. QoS adaptation

The model under which we are working does not support any resource reservation. As we are delivering video over the Internet we cannot guarantee that bandwidth availability is not going to change during the duration of the video delivery (although, in our experience, available bandwidth does not vary widely over the duration of a playout). In order to deal with this we have developed a simple QoS adaptation mechanism. If the available bandwidth decreases, the server will degrade to the next QoS level as determined by the degradation path. This change will be triggered by the violation of a low level QoS parameter threshold such as packet loss. After a number of experiments with various values, we have defined a loss rate of 10% beyond which the current QoS level would not be acceptable to the user. At this point the video will switch to the next level in the degradation path. The value of 10% was obtained through experimental evaluation [6].

#### 5. Stream protocol

The video and audio streams use RTP (Real-time Transport Protocol) [14] over UDP to send data. In order to perform the QoS adaptation we need to send the receiver loss rates to the server. This is done using the RTP control protocol (RTCP) [14] over UDP. One type of control packet defined in the protocol is the Receiver Report (RR). The RR packets are used to send reception quality information back to the sender. These reports include the number of packets received by the client as well as the number of packets expected (derived from the sequence number of the last packet received). Receiver reports are sent every one second by the CM client. Using this information the server is able to compute the loss rate at the client and perform the necessary adaptation.

The stream protocol is interoperable with *vic*, since we are using the *vic* client software to receive and display the video stream [9].

#### 6. User interface

The user interface is implemented using HTML forms. The QoS decision engine is implemented using a CGI script written in Perl. In the current prototype all the inputs to the negotiation process are visible to the user.

The first section of the form (figure 4) enables the user to make the content selection. In the next section (figure 5) the user can express his/her preferred QoS levels and tradeoffs. The next section (figure 6) deals with resource availability. In the first part it is possible to either enter a figure for the bandwidth or let the server make an estimation. The second part deals with the hardware limitations of the client machines. The capabilities of known machine configurations are stored within a database. The form enables the user to select the appropriate type. If the type of machine is unknown then the system will adapt until the QoS level appropriate to the system hardware is reached.

Once the form has been filled and submitted the QoS engine will present the degradation path (in a format similar to table 3) and the initial starting point for the video.

The next screen shown in figure 7 is the video delivery screen. The buttons on this screen—stop and start—let the user control the video and audio. When the start button is pressed the video (and audio) server is invoked via the CGI script, which then delivers the video at the QoS level determined by the initial starting point. As previously stated, no QoS control of the audio is performed at present. The audio presents no user interface. The video client is based on *vic* and the audio client on *vat* [9, 3]. The display has been augmented to display the current QoS level.

In the current prototype we have not integrated the video client into the browser. Although we can extend the Mosaic source code to display video in-line, our first version opted for a browser-source-code independent solution. Thus we project the video window into the browser.

#### 7. Implementation and evaluation

In this section we discuss and evaluate some of our implementation experiences. This highlights some deficiencies in the current prototype and suggests further work.

##### 7.1. World Wide Web aspects

There are two major aspects to embedding our QoS model of video and audio within the WWW environment:

- the overall architecture
- in-lining of video and audio

**FastWeb Video News Service**

News Clips:

Video. Port:   Audio. Port:

Local Host:

Figure 4. User interface—content selection.

**Preferred Quality of Service**

Frame rate	Size	Resolution
<input type="text" value="25 frame/s"/>	<input type="text" value="large"/>	<input type="text" value="medium"/>

**Tradeoffs**

In event your desired QoS cannot be met, state in order of preference which of the above QoS parameters are more important.

Frame Rate	Size	Resolution
<input type="text" value="v"/>	<input type="text" value="u"/>	<input type="text" value="v"/>
<input type="text" value="u"/>	<input type="text" value="v"/>	<input type="text" value="v"/>
<input type="text" value="v"/>	<input type="text" value="v"/>	<input type="text" value="u"/>

Figure 5. User interface—QoS selection.

**Resource Availability**

**Network**

Estimate available bandwidth?

Yes ( Warning! This may take time )

No      Bandwidth:  K/s

---

**Client**

CPU	Display	Video Hardware
<input type="text" value="Alpha"/>	<input type="text" value="24 bit Color"/>	<input type="text" value="JPEG"/>

Figure 6. User interface—resource availability.

**7.1.1. Architecture.** For our initial prototype we have chosen the architecture shown in figure 1. In order to better integrate the QoS architecture into the WWW, two approaches could be taken. In the first approach the HTTP protocol could be extended to perform QoS negotiation as well as deliver real-time streams. The second approach exploits Java and CORBA.

If we consider the first approach, two aspects need to be considered. The first is how HTTP can be extended to perform QoS negotiation. In the HTTP1.1 draft standard an

optional feature for content negotiation is introduced. This allows the client to assign weighting to available variants of a requested resource. It should be possible to extend this feature to perform QoS negotiation.

Another requirement is how the real-time protocol required to deliver the video/audio stream can be handled within HTTP. This issue is addressed in proposals for the HyperText Transfer Protocol—Next Generation (HTTP-NG) [17]. This suggests changing the protocol model of HTTP, which uses a separate connection for each request,

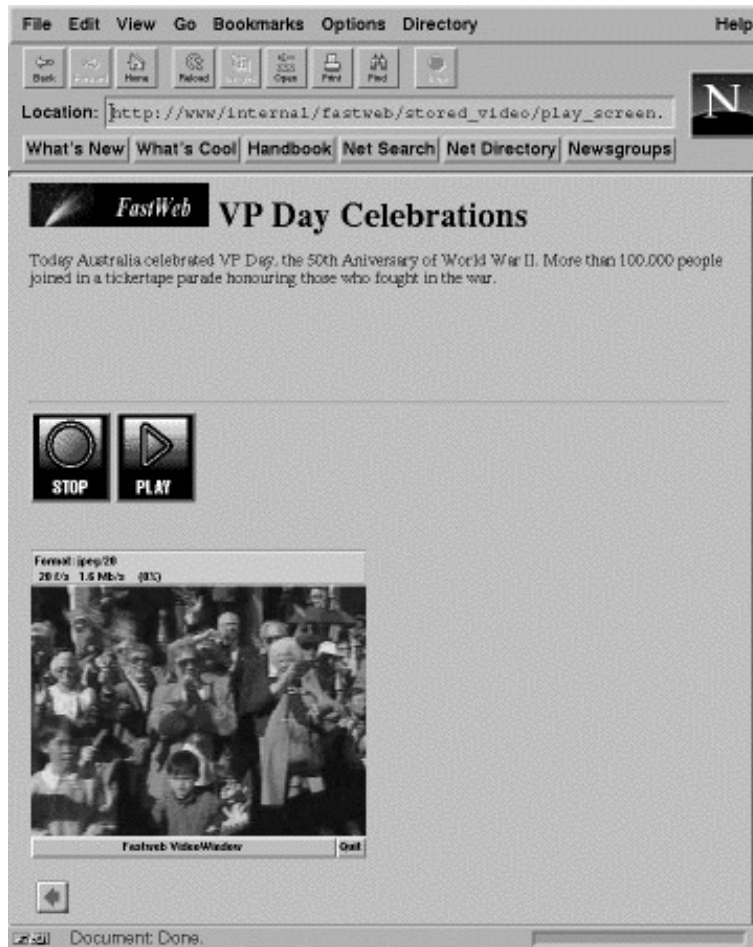


Figure 7. Video control screen.

to one which uses a single connection for many different requests, with virtual channels for various requested objects and one channel for control. If a special purpose transport optimized for a particular type of media (e.g. video) is required, HTTP-NG refers the user to this service for the actual data transfer while still handling the control information.

In the second approach (our preferred approach) we intend to exploit Java to extend functionality and supported protocols within a browser. We are particularly interested in an open and object-oriented approach.

Programs written in the Java programming language can be down-loaded on the fly as components of WWW pages. Such programs are actually represented in an intermediate form called byte-code. A Java-enabled browser executes these programs by interpreting the byte-code at run-time. Thus the programs (applets) execute within a Java virtual machine that provides protected access to client resources. The use of Java has many potential benefits. Executables are seamlessly portable across all platforms supporting a Java virtual machine. Furthermore, the client should receive the most recent version of any applet.

The Object Management Group (OMG) is in the process of adopting a Java language binding for CORBA

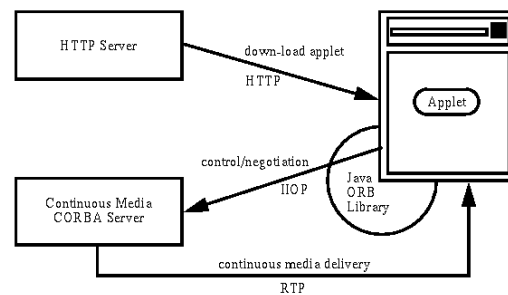


Figure 8. CORBA/JAVA system architecture.

[10]. Object Request Brokers (ORBs) which support this language binding are also known as Java ORBs. Initial implementations of such ORBs are already available from multiple vendors [20].

This approach allows us, on the one hand, to implement CORBA based servers in any of the programming languages supported by CORBA, including C, C++, Ada and Smalltalk. On the other hand, clients can be implemented as Java applets and can be down-loaded in a WWW browser. figure 8 depicts the system's architecture

based on Java and CORBA.

The advantage of this approach is that client and server can communicate directly:

- through OMG IDL defined interfaces
- avoiding the HTTP server and the CGI
- using CORBA's more efficient Internet Interoperability Protocol (IIOP)

Additionally, this approach can be extended by experimenting with various approaches to CORBA stream interfaces [12]. This includes the delivery of continuous media as well as the QoS control of the streams.

**7.1.2. In-lining.** For our prototype we needed to in-line audio and video into the browser window. The outcome presented here is not a proper integration of the windows. Two long term solutions to the problem are available. The first is *plugs-ins* in Netscape 2.0. However, at the time of writing plug-ins were yet to be released under Unix. The other solution would be to write the video client using Java. However the current implementation of Java has limited performance which would make it incapable of rendering video frames at the speeds required. It is anticipated that these problems will be overcome in the future with further optimization of Java libraries and the advent of just-in-time compilers which would speed up the execution of Java code. The Java solution has the added advantage of platform independence over the plug-in solution.

Since neither of the above solutions is available, as an interim solution we are modifying the Mosaic browser (for which source code is available) to include video and audio. Mosaic is written using the Motif toolkit, while the original video and audio clients were written using the Tcl/Tk toolkits. In order to integrate the two it would have been necessary to rewrite the user-interface of the audio/video using Motif. However using an extension to Tcl/Tk known as CTAXT (**Combine TcltkwithArbitraryXToolkits**) it was possible to integrate them with very little modification to the original code.

## 7.2. QoS management

In our model, the level of end-to-end QoS is determined by a number of resourcing factors within the internetwork and the client end system.

At the network level we have adopted a simple probing technique to measure transient network resources. We then use an adaptation mechanism to cope with any changes in network resources during the real-time payout.

Given the shared, non-deterministic nature of current IP networks, it is difficult to envisage any better technique for estimating resource availability. In the longer term it is our view that distributed applications (at least in some restricted domains, such as intranets) will be able to negotiate for more deterministic network performance. This is supported by trends in ATM networking and by work on integrated services for the Internet [8]. We are currently migrating our prototype to a local ATM environment in order to integrate the notion of network resource reservation.

To determine client resources we have initially taken a fairly simplistic approach whereby we 'know' about various combinations of resource factors (CPU speed, frame buffer type, decompression hardware) and the upper limit that these combinations impose on delivery of the video clips. However, this solution has scaling problems. In a general, heterogeneous environment it is difficult to determine *a priori* what the capacity of a given client will be for an arbitrary stream of video. This issue requires further research. One approach has suggested the use of a Quality of Service Management Information Base (QoSMB) [5].

## 7.3. QoS model

Our model characterizes video delivery into discrete bands. This contrasts with the approach of more linear adaptation of delivery. Linear adaptation is usually based only on frame rate, although there are examples (using live video capture) that try to also adapt picture quality [1]. Our model in principle permits adaptation using any aspect of perceived QoS level, many of which are inherently discrete, such as picture size and colour.

The value of the user preferences, and the consequent degradation path, needs to be assessed. The question to be answered is how well this improves information delivery. Some qualitative and/or quantitative user testing is required to evaluate this and other aspects of our QoS model.

Nonetheless the provision of adaptive delivery requires a scalable representation of information. This problem is discussed below.

Finally, we only ever degrade QoS level whenever packet loss exceeds a threshold. A more complete model would possibly permit the upgrade of the QoS level whenever resource availability improves. Using a deterministic network this would imply some periodic requests for extra network resources. In an IP network the application would need to attempt regularly to transmit at the next higher step in the degradation path. This raises the issue of the appropriate period, and a potential stability problem as the system oscillates between levels of service.

## 7.4. Scalability

Our initial approach to discrete levels of video delivery does not scale. In our first prototype we actually store multiple copies of the video clips (our priority was to produce a demonstrable version of our QoS model). This is clearly not a generic or practical solution.

A proper solution requires inherent scalability properties in the video encoding method. Ideally, we would like to store a single copy of our video (at a cost not significantly greater than existing, non-scalable schemes), and have the ability to extract whatever level of information is appropriate to available delivery resources. We are doubtful that existing, DCT-based schemes have the properties of scalability that we desire. We are therefore investigating wavelet-based encodings, with particular focus on scalability of frame rate and picture quality.

## 7.5. Audio/video synchronization

We are working on providing inter-stream synchronization of audio and video. The audio and video are received by the client as two separate streams. Each packet has a RTP timestamp associated with it. The timestamp provides a (media-specific) offset into the clip. A simple algorithm matches the timestamps for related audio and video packets. In order to achieve synchronization, we could simply decode related packets, buffer them, and then playout the buffered frames according to the timestamp information. However there are practical problems associated with buffering many uncompressed frames of video due to their size (e.g. an uncompressed NTSC frame is approximately 200 Kbytes). To avoid this we need to buffer video packets before they are decompressed. Each packet is buffered until its playout time minus the time needed to decompress and render it. However decompression time is variable and content dependent. Thus the estimation of this variable time determines the accuracy of the synchronization achieved. We are currently working on estimation schemes.

## 8. Related work

There are a number of commercial as well as experimental WWW systems that provide integrated animations, audio and video capabilities. Some systems with high visibility are Shockwave, RealAudio, StreamWorks, VDOLive and Vosaic. We have studied these systems in order to compare them with our work. However, in some cases there has been little documented information available and very restricted access to working systems. The following is therefore in some cases based on limited experience.

### 8.1. Shockwave

Director is a multimedia authoring tool for CD-ROM based productions. Shockwave for Director [16] has two components. On the HTTP server side the Afterburner tool compresses Director movies to make them available on the Internet. On the client side the Shockwave plug-in lets the user incorporate Director movies into the page layout of their HTML document.

The current Shockwave plug-in is not streaming, i.e. the Director movies need to be down-loaded in their entirety before playback. However Shockwave intend to release a new version where components of the movies will be streamed to the user. An interim release will allow for streaming audio and video including VDOLive, RealAudio and QuickTime (from Apple Computer Inc). It is claimed that this will be achieved by building components within Shockwave which allow control of those programs.

The current release of Shockwave allows for a separate, real-time audio stream. The bit rates of audio streams can be set to 8, 16, 32 or 62 kb s<sup>-1</sup> by the developer. The stream rate is chosen by the developer on the basis of bandwidth capabilities of likely users. The audio is then delivered at the bit rate encoded, regardless of the user's network connection. Thus there is no notion of negotiable and adaptable QoS.

### 8.2. RealAudio

RealAudio from Progressive Networks [13] provides real-time audio at rates of 14.4 kb s<sup>-1</sup> and 28.8 kb s<sup>-1</sup> (at the time of writing). RealAudio does not support video. Little information is provided about how RealAudio deals with network latencies apart from saying it uses special buffering techniques. Audio files must be recorded in different versions for the different data rates. It appears that the choice of stream (14.4 versus 28.8) is made by way of a user-selected option in the client. There is no adaptation of the audio stream.

### 8.3. StreamWorks

StreamWorks [18] claims to permit the delivery of streaming video and synchronized audio over the WWW using UDP/IP. Streams can be pre-recorded or live feeds. The video streams are MPEG1, while the audio could be MPEG1 or MPEG1 private data streams containing MPEG2 LBR audio. Providers encode content at 8.5, 24, 56 or 112 kb s<sup>-1</sup> depending on the the bandwidth capabilities of their potential users. Thus there does not appear to be a notion of negotiated QoS. However, StreamWorks supports a process called 'thinning' whereby a high bandwidth stream can be reduced to be transmitted over a lower bandwidth connection. We believe the server makes the decision to thin based on the bandwidth settings of the client software. We do not believe that this can be changed mid-stream.

### 8.4. VDOLive

Similar to StreamWorks, VDOLive from VDOnet Corp. [19] delivers streaming audio and video. However, it is claimed that content providers need only one video source, which can be scaled on the fly for both high and low speed network connections. VDOLive claims to be able to deliver between 10–15 frames per second over a 28.8 kb s<sup>-1</sup> modem using a proprietary video compression scheme based in part on wavelet techniques. VDOLive appears to be quite related to our work, but we have been unable to investigate its functionality.

### 8.5. Vosaic

Vosaic [4] has similar objectives to our work. It uses a feedback and a feedforward scheme to adapt to both network and end system conditions. However, Vosaic does not encompass end-to-end QoS management which involves the user.

## 9. Current status

In this paper we have presented our approach to delivery of continuous media in the WWW environment. This approach is unique in its particular focus on QoS. We have investigated QoS for the WWW and based on the outcome of this investigation we have defined an architecture for the delivery of continuous media in WWW. This includes a QoS negotiation protocol and a QoS adaptation scheme.

The practicality of our approach has been shown by a prototype implementation. We have discussed our implementation experiences.

The first prototype was demonstrated in October 1995. It supports video in the formats of moving JPEG and H.261. Audio and video are delivered independently, unsynchronized. In addition to stored video playback, the prototype also permits the delivery of live video captured at the server machine. This delivery is QoS controlled in the same manner as for stored video. The prototype also presents an integrated, WWW-based access interface to MBONE conferences.

The prototype has been tested over wide-area, metropolitan and local networks with bandwidths ranging from a few hundred kilobit s<sup>-1</sup> to a hundred megabit s<sup>-1</sup>.

The prototype certainly demonstrates the feasibility of controlled video delivery within a WWW environment. It also now provides a test-bed for further research. We have described some of our further research. In the near future we will be releasing the Mosaic client browser for general use and evaluation by the Internet community.

### Acknowledgments

The work reported in this paper has been funded in part by the Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of Australia. The authors would like to thank the reviewers of this paper for their helpful comments.

### References

- [1] Bolot J C and Turlitti T 1994 A rate control mechanism for packet video in the internet *Proc. IEEE INFOCOM'94 (Toronto)* pp 1216–23
- [2] Campbell A, Coulson G, Garcia F, Hutchinson D and Leopold H 1993 Integrated quality of service for multimedia communications *Proc. IEEE INFOCOM'93 (San Francisco, CA)*
- [3] Casner S and Deering S 1992 First IETF Internet audiocast *ACM Comput. Commun. Rev.* **22** 92–7
- [4] Chen Z, Tan S-M, Campbell R H and Li Y 1995 Real time video and audio in the World Wide Web *Proc. 4th Int. World Wide Web Conf. (Boston, MA)*
- [5] Cho H, Fry M, Seneviratne A and Witana V 1995 Towards a hybrid scheme for application adaptivity *Proc. COST 237 Workshop (Copenhagen, 1995)* (Lecture Notes in Computer Science) (Berlin: Springer) pp 176–90
- [6] Jang J 1996 Open distributed processing in multimedia networks *MSc Thesis* University of Technology, Sydney
- [7] Kerherve B, Vogel A, Bochmann G V, Dssouli R, Gecsei J and Hafid A 1993 On distributed multimedia presentational applications: functional and computational architecture and QoS negotiation *Proc. 5th Int. Workshop on Protocols for High-speed Networks* (London: Chapman & Hall) pp 1–17
- [8] IETF Secretariat—Corporation for National Research Initiatives 1996 *Integrated services (intserv) charter* <http://www.ietf.org/html.charters/intserv-charter.html>
- [9] McCanne S and Jacobsen V 1995 vic: a flexible framework for packet video *Proc. ACM Multimedia (San Francisco CA, 1995)* pp 511–22
- [10] Object Management Group and X/Open 1995 *The Common Object Request Broker: Architecture and Specification* Revision 2.0, Framingham, MA
- [11] Partridge C and Pink S 1995 An implementation of the revised internet stream protocol (ST-II) *Proc. 2nd Int. NOSSDAV Workshop (Heidelberg, 1995)*
- [12] Raymond I 1996 *Streams for CORBA, White paper* OMG
- [13] RealAudio 1995 Delivering time-based information over the Internet: HTTP versus real audio client–server streaming [http://www.realaudio.com/products/http\\_vs.ra.html](http://www.realaudio.com/products/http_vs.ra.html)
- [14] Schulzrinne H, Casner S, Frederick R and Jacobson V 1995 RTP: a transport protocol for real-time applications *Internet Draft*
- [15] Seneviratne A, Fry M, Witana V, Saparamadu V, Richards A and Horlait E 1994 Quality of service management for distributed multi media applications *Proc. IEEE Communication Society Phoenix Conf. on Computers and Communications (Phoenix, AZ, 1994)*
- [16] Macromedia, Inc 1996 Macro Media Shockwave Developers Center <http://www.macromedia.com/shockwave/developer.html>
- [17] Spero S 1993 Progress on HTTP-NG <http://www.w3.org/pub/www/protocols/http-ng/http-ng-status.html>
- [18] Streamworks 1995 Streamworks technical description [http://204.62.160.251/streams/info/streamwk\\_gen\\_info.html](http://204.62.160.251/streams/info/streamwk_gen_info.html)
- [19] VDOnet Corp 1996 VDOnet—real-time video and audio over the Internet <http://www.vdolive/>
- [20] Vogel A 1996 WWW and Java—Threat or challenge to CORBA *MiddlewareSpectra* Spectrum Reports, Winchester
- [21] Vogel A, Kerherve B, Bochmann G V and Gecsei J 1995 Distributed multimedia applications and quality of service—a survey *IEEE Multimedia* **2** (2) 10–9
- [22] Zhang L, Deering S, Estrin D, Shenker S and Zappala D 1993 RSVP: A new resource reservation protocol *IEEE Network Mag.* **7** (5) 8–18