# An Open Approach to Real-Time Multimedia on the World Wide Web

Michael Fry[1], Aruna Seneviratne[2], Andreas Vogel[3] and Varuni Witana[4]

email: *mike@socs.uts.edu.au, aps@ee.uts.edu.au, andreas@dstc.edu.au, varuni@dstc.uts.edu.au*

School of Computing Sciences[1], School of Electrical Engineering[2], CRC for Distributed Systems Technology[4]
University of Technology, Sydney, Sydney 2007 Australia.

CRC for Distributed Systems Technology[3]
University of Queensland, Brisbane 4072 Australia.

## 1 Introduction

We have produced an experimental prototype system for delivery of real-time audio and video within the World Wide Web (WWW) environment. The system is called SuperNOVA - Negotiated On-line Video Access (and Super because we think it so). A major purpose of SuperNOVA was to investigate the use of Quality of Service (QoS) management in the WWW. We have specified a negotiation protocol and adaptation scheme that permits the system to adapt real-time delivery according to prevailing network and end-system conditions.

SuperNOVA has been successfully implemented. However the architecture of the first SuperNOVA prototype is essentially closed. Although standard protocols (HTTP, RTP, UDP) are used for QoS negotiation and for real-time delivery, there are no exportable interfaces for the processes of content selection, QoS negotiation, stream delivery etc.

In this paper we first briefly describe our first SuperNOVA prototype. We then present an architecture based on Java and CORBA which is potentially a more open solution. We are currently implementing this new architecture. SuperNOVA version one has been fully described elsewhere [8]. In Section 2 we repeat this via a brief outline. This is for the sake of comprehensibility of this position statement. We believe the main contribution of this statement to be the proposed new architectural approach in Section 3.

## 2 Current SuperNOVA Architecture

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

- 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. 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.

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

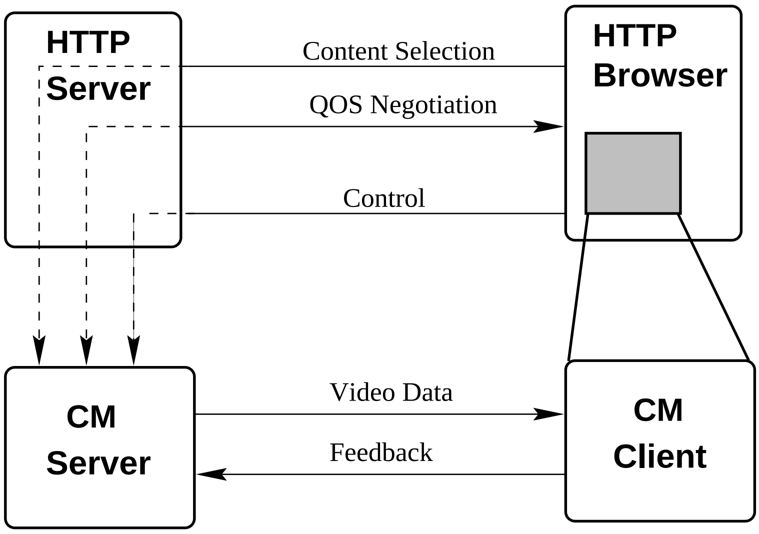


**Figure 1: Initial Architecture**

### 2.1 The QoS Management Model

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 users 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 QoS Decision engine is the heart of the QoS negotiation process. Its inputs and outputs are shown in Figure 2. These are explained in more detail in the following sections.

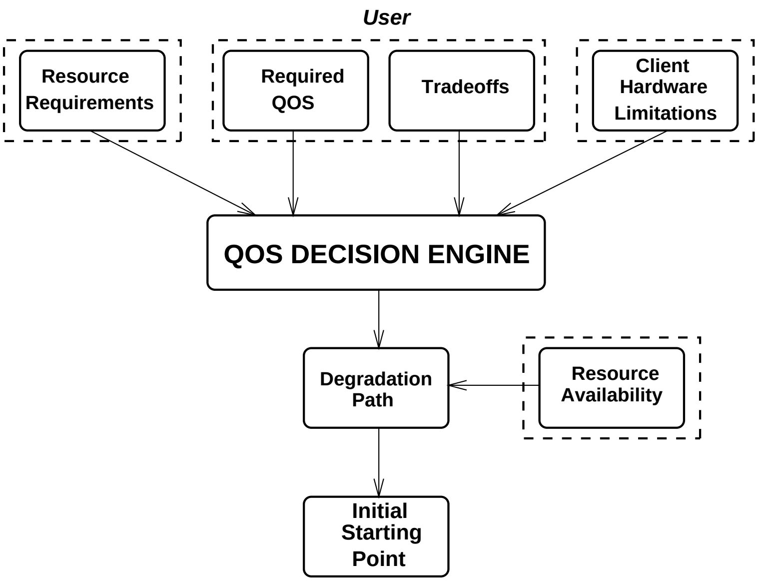


**Figure 2: QoS Decision Engine**

### 2.2 Inputs

- *Resource Requirements* - The resources needed to maintain a particular QoS level is an input to the negotiation. The only resource currently considered is network bandwidth. The necessary bandwidth

| 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 1: Bandwidth Requirements for available QoS Levels and Encodings in Kb/s**

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. We have recorded each clip in two encoding schemes (JPEG and H261) in order to study resource requirements. 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 limits the video options that can be received on a particular hardware configuration. Significant factors are availability of hardware decoding and CPU power.

### 2.3 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 users tradeoffs are.
- *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 our 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.

### 2.4 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.

### 2.5 Stream Protocol

The video and audio streams use RTP (Real-time Transport Protocol) 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)[5] 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 [2]. We use the *vat* client software to receive and play the audio [1].

### 2.6 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. Once the form has been filled and submitted the QoS engine will present the degradation path and the initial starting point for the video. The video delivery screen has a start button. When 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. The *vic* display has been augmented to display the current QoS level.

## 3 A New Architecture

Our initial prototype has many limitations. The QoS management model is simplistic. We do not implement truly scalable video, therefore the system does not scale well. However we focus on the architectural limitations here and discuss some improved approaches, one of which we are currently implementing.

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.

### 3.1 HTTPng

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) [6]. This suggests changing the protocol model of HTTP, which uses a separate connection for each request, to one which uses a a single connection for many different requests, with virtual channels for various requested objects and one channel for control. If a special purpose transport optimised 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.

Some of the authors are currently exploring approaches to HTTPng via the High Performance Protocol Architecture (HIPPARCH) project. The aim is to decouple issues of content selection, negotiation and delivery. In this case we are also encompassing negotiation over protocol functions.

### 3.2 Java and CORBA

In the second approach we intend are exploiting 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 [3]. 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 [7].

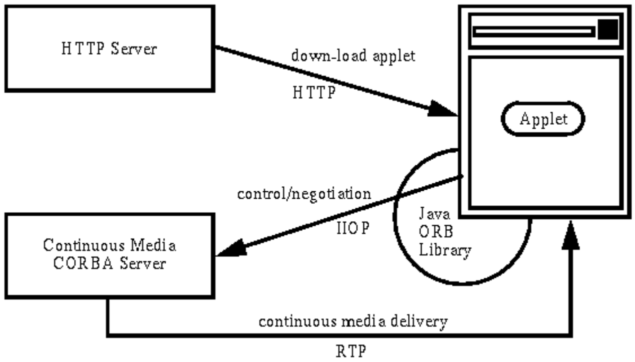


**Figure 3: CORBA/JAVA system Architecture**

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 3 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)

We are using BlackWidow as the client and ORBIX at the server. Once the client has been downloaded the protocols for content selection and QoS negotiation are represented via exported IDL definitions. This allows for potential interaction with multiple servers, that is also transparent and interoperable across multiple client types.

Additionally, this approach can be extended by experimenting with various approaches to CORBA Stream Interfaces [4]. This includes the delivery of continuous media as well as the QoS control of the streams.

During workshop presentation we will be able to provide a more detailed update on architectural implementation including our object-level design.

## 4 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 to our work. We generally find that these systems do not realise any significant form of QoS model. Furthermore we are yet to find any attempted implementations using CORBA and Java.

## Acknowledgements

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.

## References


[1] S. Casner and S. Deering. "First IETF Internet Audiocast". *"ACM Computer Communication Review"*, 22:92–97, July 1992.

[2] S. McCanne and V. Jacobsen. "vic: A Flexible Framework for Packet Video". In *"Proc. of ACM Multimedia"*, pages 511–522, San Francisco CA, Nov. 1995.

[3] "The Common Object Request Broker: Architecture and Specification". Object Management Group and X/Open, Revision 2.0, Framingham, MA, 1995.

[4] K. Raymond. "Streams for CORBA, White paper". OMG, 1996.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications", Internet Draft, 1995.

[6] S. Spero. "Progress on HTTP-NG [web page]". `http://www.w3.org/pub/WWW/Protocols/HTTP-NG/http-ng-status.html`, 1995.

[7] A. Vogel. "WWW and Java - Threat or Challenge to CORBA?". *"MiddlewareSpectra"*, May 1996.

[8] M. Fry, A. Seneviratne, V. Witana, and A. Vogel. "Delivering QoS Controlled Continuous Media on the World Wide Web". In *Proc. of IWQoS96*, Paris, March 1996.