HTML extensions for Multimedia Documents and Quality of Service management on the Web

Erika MADJA, madja@iro.umontreal.ca,
Grégor v. BOCHMANN, bochmann@iro.umontreal.ca,
Rachida DSSOULI, dssouli@iro.umontreal.ca,
Jan GECSEI, gecsei@iro.umontreal.ca

Département d'informatique et de recherche opérationnelle

Université de Montréal

(February 1997)

Abstract

The purpose of this paper is to describe an extension of HTML which supports the description of multimedia documents, including text, image, audio and video, and which includes necessary meta information for quality of service (QoS) negotiation. Based on the general structure of multimedia documents and associated QoS parameters, which where developed under an ongoing CITR project, we propose an HTML extension which uses the recently defined OBJECT and RESOURCE elements. An ongoing implementation of a user agent for QoS negotiation using these extensions is also described. This implementation intends to allow the concurrent use of presentation protocols which are currently in use over the Internet for the presentation of real-time audio and video.

1.0 Introduction

In the context of a CITR (Canadian Institute for Telecommunications Research) project, a prototype system for remote access to News-on-Demand [8] has been developed. This system allows the user to remotely access a multimedia database over ATM and other types of networks. Quality of Service (QoS) negotiation and adaptation had been taken into account. For instance, a given document may exist in different variants on different sites and possibly corresponding to different presentation qualities, such as video and audio quality, resolution etc. According to the given user preferences and the target system capabilities, the QoS negotiation and adaptation feature allows for the selection of the best configuration for the user and for automatic adaptation in case of changes such as network congestion. The News-on-Demand prototype is an integration of different software components developed by the various sub-groups of the CITR collaboration: The distributed multimedia database (DBMS) from University of Alberta, the continuous media file server (CMFS) from University of British Columbia, a synchronization component from University of Ottawa and a QoS Module from University of Montréal.

The aim of our present work is to produce a version of the CITR prototype compatible with the Web. Basically, the idea is to be able to choose between different media variants from different sites depending on the user specified preferences and the system capabilities. The project is motivated by the fact that the Web and its associated set of protocols do not support adequately the real-time transfer of the continuous media (Audio/Video). This inadequacy is due to the fact that the underlying Internet transport protocol (TCP/IP) does not perform resource reservation for real-time data traffic. Sometimes, Audio/Video clips data are completely downloaded before being played back by an appropriate application. This scheme becomes easily impractical for long video clips. Our project foresee access, through the Web, to multimedia documents that are distributed over heterogenous servers like the http servers (for texte and image components) and the Continuous Media file servers (CMFS) that was used in our initial prototype. Other servers such as Vosaic and RealAudio/RealVideo will also be considered for providing components of a multimedia document.The system provides a user interface for the QoS negotiation which enables the

user to specify his preferred media presentation quality. The Audio/Video component will be included in the HTML pages as a Java applet (a button on which the user click to actually access the audio/video component). When a user reaches such applets, his preferences and the accessibility of the different media variants are analyzed first, and then an appropriate variant is presented to him.
To be able to support the Quality of Service (QoS) function on the Web, it is necessary to define an extension for the HTML language conveying the media metadata related to QoS issues. Such an extension is described in the following.

2.0 Multimedia document structure and quality of service

In our CITR project, we have defined a general logical structure of multimedia documents, which typically are composed of one or more monomedia components like text, images, video or audio. It allows for different variants (with different QoS parameters) for each monomedia component of a multimedia document and includes attributes describing the QoS qualities for each variant. The metadata of a multimedia document must include this information in order to allow meaningful QoS negotiation and resource allocation. The document structure and the metadata are detailed on Figure 2 in Annex 2. Some additional discussion of these concepts is also included in Annex 2.

3.0 Using the OBJECT and RESOURCE tags to describe the multimedia document structure and quality of service parameters

3.1 General comments

As stated above, metadata are necessary for the QoS management activity. As we are in the Web environment and the text and image parts of the multimedia documents are already coded in HTML, we propose to directly include the metadata in the HTML documents. For the needs of multimedia and QoS metadata we have investigated various proposed HTML extensions. We have retained two newly defined tags (<RESOURCE> and

<OBJECT>) that seem suitable for the description of the metadata and the definition of multimedia (audio and /or video) document structures in HTML. The description of the <RESOURCE> and <OBJECT> elements can be found in Annex 1. A more complete description can be found in [5] and [4].
The <OBJECT> tag allows the HTML author to specify the data and /or the properties/ parameters for initializing objects to be inserted into HTML documents as well as the code that can be used to display/manipulate that data.This definition is well suitable for our need to include multimedia object (video and/or audio) with facilities to specify the parameters (metadata) and indicate the code that handles (QoS negotiation and media displaying program) the multimedia object. We use the <RESOURCE> element to deal with the description of resource variants. **It provides information on some resource within another HTML document**. The <LINK> elements, when used with their REL attribute enable the description of the resource variants. The resource is then a generic resource while variants are specific resources. Each <LINK> element describes a resource variant. The relationships of the <LINK> element provide a mean to indicate how resource variants differ from one another i.e the link type provides a mean to specify what parameter of quality varies.

3.2 Describing QoS parameters and multimedia document structure

In the case of our application, the multimedia objects to be inserted are the video and/or audio components of a multimedia document. The code that implements the object includes a QoS negotiation module and the media display module. The metadata, i.e the QoS parameters are used to “initialize” the object. Our metadata describes the multimedia object as a list of monomedia objects, each monomedia having a number of variants and each variant having in turn its own QoS characteristics. We have used <RESOURCE> to code the metadata. Since our metadata are application specific, we have used their experimental format, which means that parameter names are preceded by “x-”. For the description of the resource variants, none of the existing relationship seems to fit our need. In fact, the variants may differ by a combination of QoS parameters instead of a single QoS parameter. So, we propose a “QoS-specific” relationship to simply indicate that variants have several QoS characteristics. Giving these semantics, we can now describe meta-information on our multimedia objects and insert them into a HTML document. Suppose the multimedia document is composed of a text part coded in HTML, an video having only one variant and an audio available in two variants. We propose the following description of the metadata for the audio and the video in HTML:

```
<resource href = Audio_monomedia>
<link rel = QoS-specific    href = audio1>
<link rel = QoS-specific    href = audio2>
</resource>
<resource href = audio1>
<meta   http-equiv = url value =” pnm://audio.realaudio.com/audiofile1.ra”>
<meta   http-equiv = x-format        value = g728>
<meta   http-equiv =   x-duration    value = 16>
<meta   http-equiv = x-quality   value = phone>
</resource>
<resource href = audio2>
<meta   http-equiv = url value = ” pnm://audio.realaudio.com/audiofile2.ra”>
<meta   http-equiv = x-format        value = g728>
<meta   http-equiv =  x- duration    value = 20>
<meta   http-equiv = x-quality   value = cd>
</resource>
<resource href = video>
<meta   http-equiv = url value = “cmfs://sitename/UIO-texte-file”>
<meta   http-equiv = x-format        value = h261>
<meta   http-equiv =   x-duration    value = 10>
<meta   http-equiv = x-framerate   value =30 >
</resource>
```

To insert the multimedia object in HTML,

```
<OBJECT id = “multimedia-object” classid = "negotiation-prg">
 <PARAM name = audioComponent value = "Audio-monomedia">
 <PARAM name = videoComponent value = "video">
 </OBJECT>.
```

4.0 Towards implementation

4.1 A Java-based user agent for QoS negotiation

Based on our experience with the CITR prototype, we are implementing a user agent which understands the HTML-encoded metadata described in Section 3. We propose a Java solution. Before the access, transfer and display of the video, some initialization activities must take place that include:

1- to get the metadata

2- to get the user QoS preferences via an user interface or use an existing user profile.

3- to perform QoS negotiation in order to choose the best variant for the user that can be supported by the system.

4- If such a variant exists, display it.

As <OBJECT> and <RESOURCE> are not yet supported, we activate the object via an Java applet that receives as parameter, the identifier (id) of the object. The applet then finds the object and associated metadata in the extended HTML page and performs the necessary processing. The metadata must be in the same HTML document with the applet.

The user can specify his acceptable values for the QoS parameters via a graphical user interface. For video, we have retained three QoS parameters at the user level. They are: frame rate, resolution and color. For audio, we have quality (Phone or CD) and language. The cost is also provided as a choice criteria. The user may also define his priority among the QoS parameters, on one hand, and on the two monomedia, on the other hand. As an example of QoS specification by the user, we consider the following:

FrameRate >=10

Resolution >= medium

Cost < 1$

Priority1 = Cost

Priority2 = FrameRate

Priority3 = Resolution

The priority system enables classification among variants. Giving a user profile and the available variants, the QoS negotiation engine chooses the best variant that the system can support at the given time. In case of a problem, an adaptation will be made by considering the next best variant.

4.2 Integration of various servers

The original version of our the News-on-Demand system uses the CMFS server for accessing and retrieving continuous media raw data. Our Web version will also provides this facility.

However, we also intend to include other continuous media servers existing in the Web environment: The Vosaic server [6], the RealAudio/RealVideo servers[7].

Vosaic is a client-server system developed at the University of Illinois that provides real-time video on the Web. It uses the VDP (Video Datagram Protocol) protocol for continuous media transfer. VDP uses an adaptation algorithm to find the optimal transfer bandwidth. RealAudio developed by Progressive Networks, provides high quality audio over the Web. Progressive Network announces recently their RealVideo System.

It would be interesting to integrate these various servers such that different media variants may use different communication protocols for real-time presentation, corresponding to the different technologies. The non continuous parts of the multimedia document could be provided by the existing HTML servers. The client software will consequently be an integration of the client software for each of these different servers.

4.3 System architecture

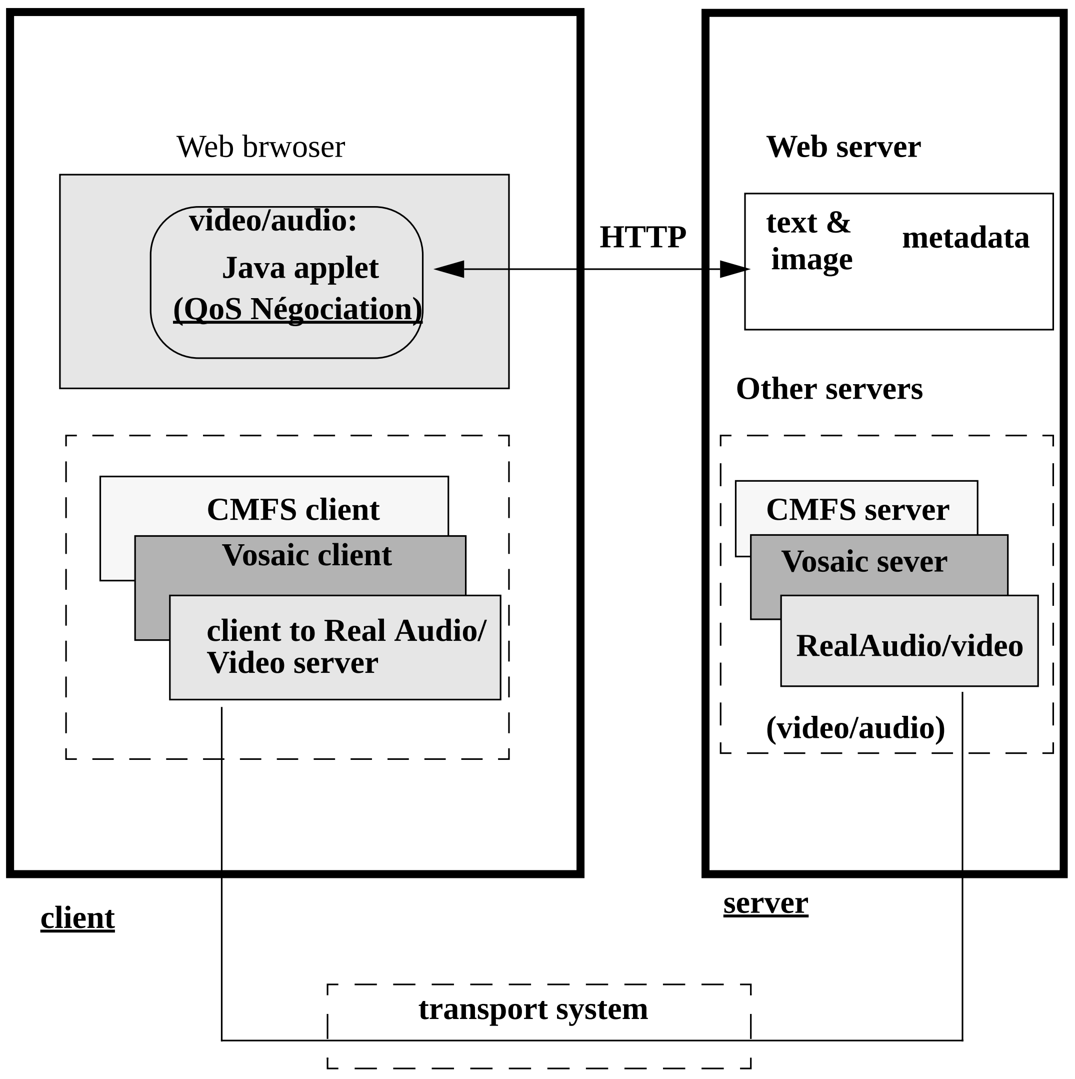

Figure 1 **Architecture of the news-on-demand system for the Web**

5.0 Conclusion

The aim of this project is to produce is Web-compatible version of our News-on-Demand prototype. The project has mainly used the semantic defined for multimedia documents, metadata on these documents as well as some of the major QoS negotiation principles of the original system. One of our preoccupations was to find the HTML extension conveying the metadata related to QoS issues. Fortunately, we didn't have to define new HTML tags. <RESOURCE> and <OBJECT> seem suitable for our need. Before they can be supported by the popular browsers, we've are implemented a Java applet which handles it. Currently we are still working on the client software for the CMFS server. There are a number of others working groups that investigate on real-time audio and video for the Web. But a QoS negotiation that involves the user was not addressed apart from the Fast Web project [10] that is similar to ours.

References

[1] Helper Applications, http://www.netscape.com/assist/helper_apps/

[2] Plug-in guide, http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/pguide.htm

[3] Component Object Model Specification, http://www.microsoft.com/intdev/sdk/

[4] Inserting objects into HTML, http://www.w3.org/pub/WWW/TR/WD-object.html

[5] Giving Information About Other Resources in HTML, http://www.w3.org/pub/WWW/MarkUp/Resource/Specification

[6] Vosaic, Continuous Media on the Web, http://choices.cs.uiuc.edu/Vosaic/Vosaic.html

[7] RealAudio, http://www.realaudio.com/ , http://www.real.com/ .

[8] Abdelhakim Hafid, Quality of service adaptation in distributed multimedia applications , August 1996, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal," ftp://ftp.iro.umontreal.ca:/pub/teleinfo/TRs/P1045.ps.gz"

[9] J. Rambaugh & al. Object Oriented modeling and Design, Prentice Hall, 1991

[10] Michael Fry, Aruna Seneviratne, Andreas Vogel and Varuni Witana "Delivering QoS Controlled Continuous Media on the World Wide Web", In Proc of IWQoS96, Paris, March 1996

Annex 1

The <OBJECT> element[4]

<u>Description</u>

<OBJECT> is an HTML extension that supports the insertion multimedia objects in HTML documents.These multimedia objects include java applets, Microsoft Component Object Model (COM) and a wide range of other media plug-ins. The <OBJECT> tag is a generalization of the IMG tag for dealing with new media. <OBJECT> allows the HTML author to specify the data and /or the properties/parameters for initializing objects to be inserted into HTML documents as well as the code that can be used to display/manipulate that data.This definition fits well our need to include multimedia object (video and/or audio) with facilities to specify the parameters (metadata) and indicate the code that handles (QoS negotiation and media displaying) the multimedia object. However, the architectural and application programming interface issues for how objects communicate with the document and other objects on the same page hadn't been covered in the current specification of the <OBJECT> tag. It is anticipated that future specifications will cover these topics. On the other hand, it doesn't provide enough flexibility to enable a rich description of resources variants.

<u>Syntax</u>

The <OBJECT> element requires both start and end tags. It has the same content model as the HTML BODY element, except that one or more optional PARAM elements can be placed immediately after the OBJECT start tag and used to initialize the inserted object. OBJECT has the following attributes:

ID

Used to define a document-wide identifier. This can be used for naming positions within documents for use as destinations of hypertext links. An ID attribute value is an SGML NAME token. It may also be used by the user agent or objects in the document to find and communicate with other objects embedded in the document.

DECLARE

Used to indicate that the object is to be declared but not instantiated.

CLASSID

This is a URL that identifies an implementation for the object. In some object systems this is a class identifier.

CODEBASE

Some URL schemes used to identify implementations require an additional URL to find the implementation. CODEBASE allows you to specify that URL.

DATA

This is a URL pointing to the object's data, for instance a GIF file for an image. In the absence of the CLASSID attribute, the media type of the data is used to determine a default value for the CLASSID attribute. The implementation is then loaded as if the CLASSID attribute had been given explicitly.

TYPE

This specifies the Internet Media Type for the data referenced by the DATA attribute in advance of actually retrieving it. In the absence of the CLASSID attribute, this allows the user agent to retrieve the code implementing the object concurrently with the data, and to skip over unsupported media types without needing to make a network access.

CODETYPE

This specifies the Internet Media Type of the code referenced by the CLASSID attribute in advance of actually retrieving it. User agents may use the value of the CODETYPE attribute to skip over unsupported media types without needing to make a
network access.

<OBJECT> has others attributes such as WIDTH, HEIGHT etc. which control the display of the object on the screen.

The <PARAM> element is used to specify a list of named property values needed to initialize the object. It has a NAME attribute which defines the property name and a VALUE attribute which specifies the property value.

Example

```
<OBJECT ID =   obj1 CLASSID = code_name>
<PARAM NAME = param1    VALUE = value1>
```

<PARAM NAME = param2   VALUE = value2>
</OBJECT>

The <RESOURCE> element

Description

the <RESOURCE> element provides a simple way to describe any resource that can de identified with a URL. **It enables information to be giving on some resource inside another HTML document**. Typically, It can be used to give metadata (title, size, author etc.) on an document destined to included in another HTML document or to dive a rich description of resource variants. It is similar to the HEAD element which convey metadata on the document itself.

Syntax

The <RESOURCE> element also requires an start and end tags. The <LINK> elements when used with their REL attributes enables the description of the resource variants. The resource is then generic resource while variants are specific resources. The <META> element gives meta-information on the resource such as document author, keywords etc. Such information can be extracted by servers/clients for use in identifying, indexing and cataloging specialized document meta-information. <RESOURCE> has the following attributes:

ID: specify the resource identifier

HREF: specify the URL address of the resource

<LINK> has the following syntax: <LINK   rel = relationship href = URL address of the variant>. The “REL” of <LINK> indicates that the resource is a generic one. Each <LINK> element describes one variant of the resource, which address is indicates by the HREF attribute. Different relationships are possible between a generic resource and its variants. The “*content-type-specific*” relationship indicates that variants have different MIME types., “*content-language-specific*” relationship indicates that the document exists in different content language (French, English). as in the following examples:

<resource   href = myimage>

<link rel = "content-type-specific"   href = "myimage.png">

```
<link rel = "content-type-specific"   href = "myimage.gif">
<link rel = "content-type-specific"   href = "myimage.avi">
</resource>
```

This RESOURCE element indicates that myimage is available in three specific content-types, with specific URLS myimage.png, myimage.gif, and myimage.avi.

```
<resource   href = mydoc>
<link rel = "content-language-specific"   href = "mydoc.eng">
<link rel = "content-language-specific"   href = "mydoc.fr">
</resource>
```

The “*choice-specific*” relationship is provided to let the choice of the variants at the discretion of the reader. In the example above, two style sheets are available for a given document, each provided by the author who leaves the choice at the reader's discretion.

```
 <resource url="mystyle">
     <title>Style options for the Kooltown telegraph</title>
     <link rel=choice-specific href="mystyle-old.css"
        title="Original Kooltown telegraph Style">
     <link rel=choice-specific href="mystyle-new.css"
        title="New Kooltown telegraph style">
   </resource>
```

The user agent might for example prompt the user with a question as to which style is preferred. Generally, relationship indicates the characteristics for which resource variants are different while choice-specific suggest the user intervention.

Annex 2

Helper Applications [1]

Helper Applications or External Viewers are the external programs used by the Web browsers to display the different types of media document. These tools interpret data like audio and video that a normal browser do not support. Each Helper Application has an associated MIME type and particular assigned file extension. Each document containing a media of that MIME type should have the corresponding extension. Web browser, like Netscape, proceed by activating the appropriate Helper depending on the file extension of the document matched during the browsing.

Plug-ins [2]

A Plug-in is a specific code for a particular platform that is designed to extend Netscape Navigator to include a wide range of interactive and multimedia capabilities. Plug-ins are associated with one or more MIME types used by the browser to activate them. When activated, the browser enumerates the available plug-ins on the platform, and registers each plug-in with the corresponding MIME type. An instance of a Plug-in is activated each time a corresponding data MIMIE type in encountered. This instance is killed when the user quit the corresponding page or when he close the holding window. Plug-ins communicate with Netscape via an API (Application Programming Interface). The main constraint associated with Plug-ins and Helper Applications is that they should reside on the client machine.

Quality Of Service (QOS)

Quality of Service is the collective effect of service performance which determine the degree of a satisfaction of the user of the service. It is described in terms of a set of userperceived characteristics of the performance of a service. In a distributed multimedia application context, Quality of Service may be considered as the set of quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application[8]. For example, the QoS parameters considered in the processing of the isochronous data are: transit delay, jitter which represents the variation of the transit delay, and the data loss rate etc.

MetaData

The metadata contains informations related to the different system components that affect the quality of service management activity. In the CITR project context, the relevant metadata for the current prototype describe the composition of the multimedia document and the characteristics of the different variants of each of its monomedia. Metadata of a video document for example, will contain the list of the variant of the video, and for each variant, the values of the QoS parameters (delay, resolution, coding format, etc.) that characterize it. Metadata could also be specified for the system components, the network, the decoder, etc. For example, metadata of a client station could be the type of the screen, decoder capacity, etc. The metadata of the network contain informations like:
- Delay: the delay introduced by the network to transport a data unit between a source and a destination.
- Jitter: the delay variation between two data units over the same connection.
- Data Rate: the number of data unit transferred per time unit
- Cost: the cost of using the network per time unit or data unit.

Multimedia Document

In the context of our project, we have defined a multimedia document as being typically composed of one or more monomedia document like text, image, video or audio. The following figure describe the structure of a multimedia document using the OMT model notations [9].

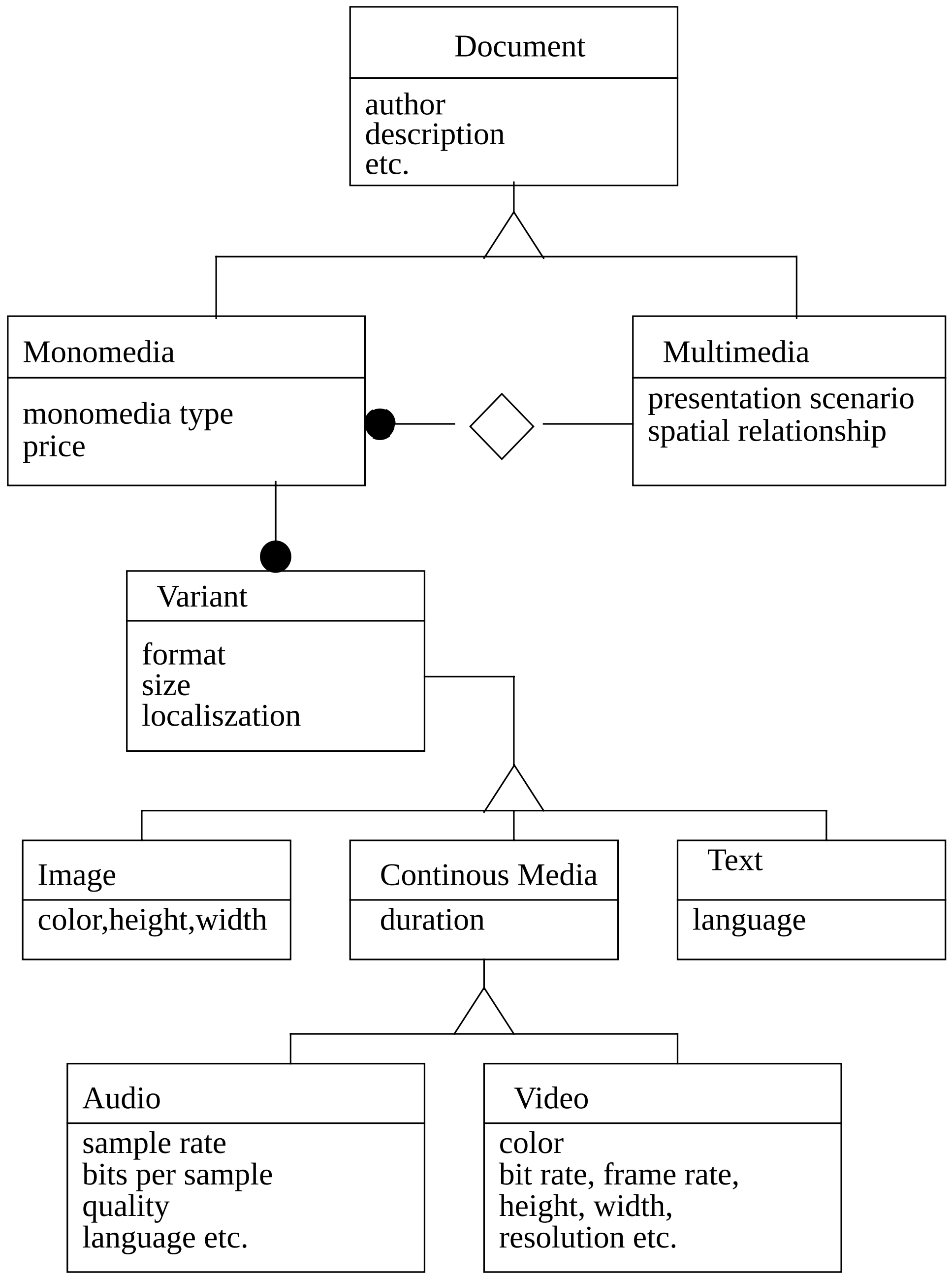

Figure 2 Multimedia document structure description

Annex 3

An adhoc integration of the News-on-Demand prototype with the Web

The notion of Helper Application was used as a first solution. It consists of using the initial CITR prototype as a Helper Application. The basic idea used in this solution is to code texte and image component of the multimedia documents in HTML and store metadata on continuous media in a simple ASCII file. Note that in the original prototype, the QoS module gets the metadata from the distributed multimedia database (DBMS) which also contains text and image files. The video and audio raw data are still on the continuous media file server. Video and audio are included in HTML as hypertext links which point to the related metadata. An experimental MIME type with the corresponding “meta” extension is associated to the metadata. The user, when clicking such a link, activates the QoS negotiation module of the CITR prototype. The program gets the metadata file name as argument instead of getting them from the multimedia database. From this moment, the web version of the News-on-Demand system behaves like the original version.

High level architecture

The client software is globally composed of a Web browser (Netscape), the QoS module, a synchronization module which accesses the CMFS server. The server part is made of the HTTP server and the CMFS server. Below is an overview of the system architecture.

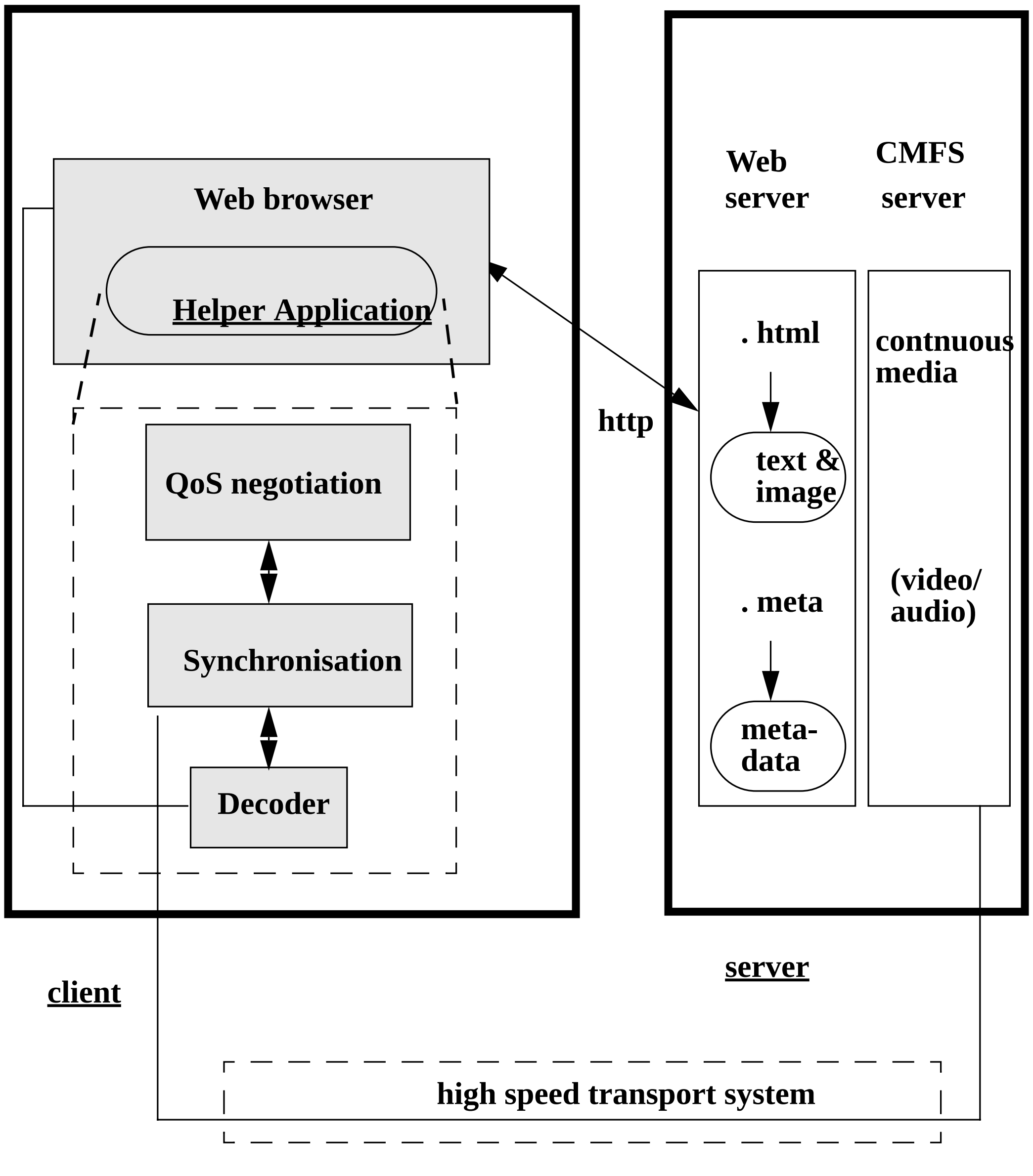

Figure 3 Architecture of the news-on-demand system implemented as helper application