# Architectural Design of Adaptive Distributed Multimedia Systems

Gregor v. Bochmann†, Brigitte Kerhervé‡, Abdelhakim Hafid†,
Petre Dini†, Anne Pons‡,

†Université de Montréal
Département IRO
CP 6128, succursale centre ville
Montréal, H3C 3J7, Canada
E-mail:{bochmann,hafid,dini}@iro.umontreal.ca
Tel: (514) 343 74 84; Fax: (514) 343 58 34

‡Université du Québec à Montréal
Département Informatique
CP 8888, succursale centre ville
Montréal, H3C 3P8, Canada
E-mail:{Kerherve.Brigitte,Pons.Anne}@uqam.ca
Tel: (514) 987 67 16; Fax: (514) 987 84 77

## Abstract

*In the context of a collaborative research project funded by the Canadian Institute for Telecommunications Research (CITR)*[1]*, we have developed a prototype system for remote access to News-on-Demand. This system allows the user to remotely access a multimedia database, containing news clips in the form of multimedia documents, over ATM and other types of networks. Special attention is given to quality of service (QoS) negotiation and adaptation. For instance, a given document may exist in different versions on different sites and possibly corresponding to different presentation qualities, such as video and audio quality, size of display and cost. A graphical interface is available for the user to select his preferences and provides the possibility of obtaining examples of specific quality features. The QoS negotiation and adaptation features allow for the selection of the best configuration for a given user request and for automatic adaptation in case of changes to the system parameters, such as network or server congestion.*

*We present in this paper an abstract architectural design of our adaptive distributed multimedia system for remote access to multimedia databases. This design focuses on the aspects of the system which are essential for QoS negotiation and adaptation, which is our main concern in the ongoing CITR research project. The paper gives a functional overview of the system and details its structural and behavioral aspects. An abstract application programming interface (API) is given and issues related to the transition from the abstract system design to an implementation are discussed.*

[1] This work was supported by a grant from the Canadian Institute for Telecommunication Research (CITR), under the Network of Centers for Excellence Program of the Canadian Government.

## 1. Introduction

Multimedia information systems integrate diverse media such as text, video and images to enable a range of multimedia applications including information retrieval. In the collaborative research project "Broadband Services" funded by the Canadian Institute for Telecommunications Research (CITR), we have taken the "News-on-Demand" application as the target for our prototype development. This target application embodies many features that are generally applicable to many multimedia presentational applications, such as digital libraries or computer-assisted training.

Our News-on-Demand prototype is an integration of software components developed by the various sub-groups of the CITR collaboration [13]. The prototype consists of the distributed multimedia database (DBMS) from University of Alberta [10] , a distributed continuous media file (CMFS) server from University of British Columbia [9] , a synchronization component from University of Ottawa [7] , and the QoS management module from Université de Montréal [6] . Scalable video encoding is studied at INRS Telecommunications [1] . With the current prototype, the user may choose a document in the database for presentation, and select the desired quality of service (QoS) including such parameters as video and audio quality, size of display, and cost. A graphical interface is available for this purpose which includes the possibility of obtaining example of specific quality features. The transmission of the continuous media components of the document, e.g. video and audio, proceeds in real-time over ATM or a local network during the presentation of the document. The system allows for several versions of a given media component, possibly with different QoS parameters and accessible over different networks. The QoS negotiation and adaptation features allow for the

selection of the best configuration for a given user request and for automatic adaptation in case of changes of the QoS system parameters, such as in case of network or server congestion.

We present in this paper a new architectural design of our system, at a relatively abstract level. As mentioned in [11] ,it is sometimes better to describe a high-level specification of a new system after the experience gained during the implementation process through the encounters of various problems and the study of possible solutions. The architectural design presented here results of our work with the News-on-Demand prototype, but it is conceived to be much more general in nature. We concentrate on those aspects of the system design which are essential to QoS negotiation and adaptation, which is our main concern within the CITR project.

The paper is organized as follows. In Section 2, we give a functional overview of our adaptive distributed multimedia system. In particular, a few use cases including QoS negotiation and adaptation are presented. The architectural design of the system is presented in Section 3 and 4 in an object-oriented framework. Section 3 contains a presentation of the major objects within the system, their attributes and operations. The behavioral aspects of the system are described in more detail in Section 4 by considering in more detail the use cases introduced in Section 2. Some additional objects are introduced and the role of all objects in the execution scenarios corresponding to the above use cases is explained in detail. An abstract application programming interface (API) is also given, based on the abstract architectural design. Issues related to the transition from the abstract system design to an implementation are discussed in Section 5, in particular aspects related to the distribution of the different system components, the implementation of active objects, including objects for multimedia stream processing, and the mapping of the abstract API onto a concrete interface in terms of the programming languages C and C++ used in our prototype. Some conclusions are given in Section 6.

## 2. Adaptive Distributed Multimedia System

Distributed multimedia systems should provide the user with efficient access to pertinent information with the required quality. However, since multimedia objects are voluminous and unstructured, manipulation, transfer and visualization of such objects can require a lot of resource and time. It then becomes essential to prevent unsatisfactory information delivery. For that purpose, Quality of Service (QoS) management appears as an essential function to be provided by distributed multimedia systems [4; 8] and significant contributions have been recently made in this field.

Our approach is similar to the one proposed in [8] where the architecture uses a brokerage model which incorporates QoS translation, QoS negotiation and renegotiation. In contrast to the previous approach where the system configurations considered are static, we propose a QoS management architecture that supports the dynamic choice of a configuration to support the QoS requirements of the user of a specific application [3] . We consider different system configurations and select an optimal one to provide the appropriate QoS support. Moreover, the service requested may be different from a data flow service, depending on the particular application.

The QoS function aims at controlling and guaranteeing the level of quality that the system is able to offer to the user. The role of a QoS manager is to determine and examine the possible alternatives to respond to a user's request, and among the different possibilities to choose the one which satisfy the QoS constraints expressed by the user and those supported by the different components of the system.

Integrating such a function leads to consider the user's requirements regarding the quality of service and the various constraints supported by the distributed multimedia system [14] . The user's requirements may concern system performance, the quality of information provided and the financial costs attached to document delivery. The system constraints include those attached to the client machines such as the screen type, to the server, to the transport systems as well as the characteristics of the multimedia objects.

In the framework of the CITR Broadband Services major project, our role is to design and develop methods for managing the resources needed for QoS adaptation in a distributed environment. Such methods are dedicated to support distributed multimedia applications that can adapt to changing QoS conditions of the underlying transport service and the remote information servers. In this section we give a general overview of our adaptive distributed multimedia system. We first give the general overview of this system and describe the different actors, and then, through a set of simple scenarios we give an intuitive presentation of the system behavior.

### 2.1. Functional Overview

The adaptive distributed multimedia system runs in a fully distributed architecture where multimedia data are stored at various sites, and where users can access from different places throughout the network. Figure 1 presents the functional view of our system in OMT object model notation [12] .

The database is the information provider. It can be supported by several database servers and stores multimedia data as well as metadata used to facilitate searching, transfer and delivery. The multimedia documents stored in the database are composed of several monomedia objects, linked together with spatial and temporal synchronization constraints. Several

physical representations can exist for a monomedia object, we here use the term of version, which correspond to a format version. As for example two versions of a same video sequence could offer different color qualities. Version 1 could be a super-color version of the video sequence, while version 2 could be the black and white version of the same video. A given version is stored on a specific server machine.

The search is processed on a client machine and the selected multimedia documents will be delivered there. The server machine is a machine located in the network on which the objects that compose multimedia documents are stored. A server machine can be either a database server, an image server or a continuous media server. The network physically links the different machines together.

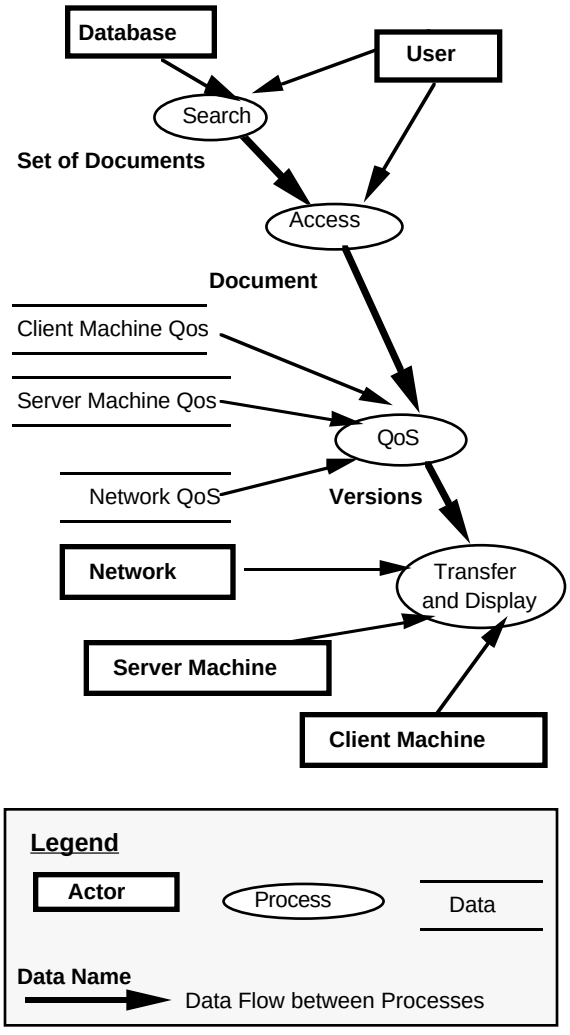


Figure 1: General Overview

For the multimedia document chosen by the user, the negotiation protocol is initiated for each composed monomedia object to select the version that matches the requirements of the user given by his user profile and the constraints of all the involved actors. These constraints are identified for each actor by analyzing its QoS information. Some of the QoS parameters, such as the client machine environment are static, while other parameters are dynamic, such as the load of the server machine or the available memory ressources. This information will be detailed in Section 3.

## 2.2. A Comprehensive Example

In this section we describe the behavior of the system while searching multimedia documents through three different scenarios.

Let's assume that our application is concerned with cultural multimedia news, and specifically with news related to movies and actors. A video sequence of the Clint Eastwood's movie "A perfect world" is stored in the database. Two versions of this video sequence exist: the super-color version is stored on server 1, and the color version is stored on server 2. Server 1 is connected to the client machine through network 1 and server 2 through network 2. Network 2 is cheaper than network 1.

Let's assume that the user's requirements concerning QoS for videos are the following: color is the minimum color-quality requested and cost should not exceed 50 cost-units. Priority between offers is given to those with lower costs. The user searches in the database for a video sequence from the Clint Eastwood's movie "A perfect world". Video sequences stored on server 1 and server 2 are relevant to the query. The following scenarios may occur at different times:

**Scenario 1:** QoS manager chooses the less expensive offer.
Transferring the super-color video sequence from server 1 to the client machine through network 1 is estimated at 45 cost-units. transferring the color video sequence from server 2 to the client machine through network 2 is estimated at 37 cost-units. The QoS manager chooses the second solution which is less expensive. This decision is made without any interaction with the user and transfer of version 1 is activated.

**Scenario 2**: the two possible offers are too expensive: negotiation is required.
Transferring the super-color video sequence from server 1 to the client machine through network 1 is estimated at 61 cost-units. Transferring the color video sequence form server 2 to the client machine through network 2 is estimated at 58 cost-units. Both offers do not satisfy the user's requirements. Thus, negotiation with the user is initiated. The two possible offers are submitted to the user who chooses the one he prefers.

**Scenario 3**: during the transfer, congestion of the network occurs.
Scenario 1 has been selected, but during the transfer of version 1 from server 1, a congestion of network 1 appears. The cost of transferring version 2 from server 2

is 45 cost-units and still lower than the user's requirement. Thus the system performs an automatic switch-over to version 2 from server 2. If the cost had been more than 50 cost-units, a re-negotiation should have been initiated with the user.

## 3. Architectural Design

In this section we present the major objects within the system: the database manager, the profile manager, the QoS manager and the network monitor. For each of these objects, we detail attributes and operations.

### 3.1. Database Manager

The conceptual object model we present now captures the semantic of a multimedia document used in the QoS negotiation protocol. We first describe the structure of the Multimedia and Monomedia entities using the OMT object model [12]. Later, we explain our choices of modeling related to the negotiation of the quality.

The multimedia and monomedia document entities that we have defined express a different semantic than what is used in classical multimedia models. They refer to logical objects collaborating in the QoS negotiation for generating a "displayable" multimedia document with the required quality.

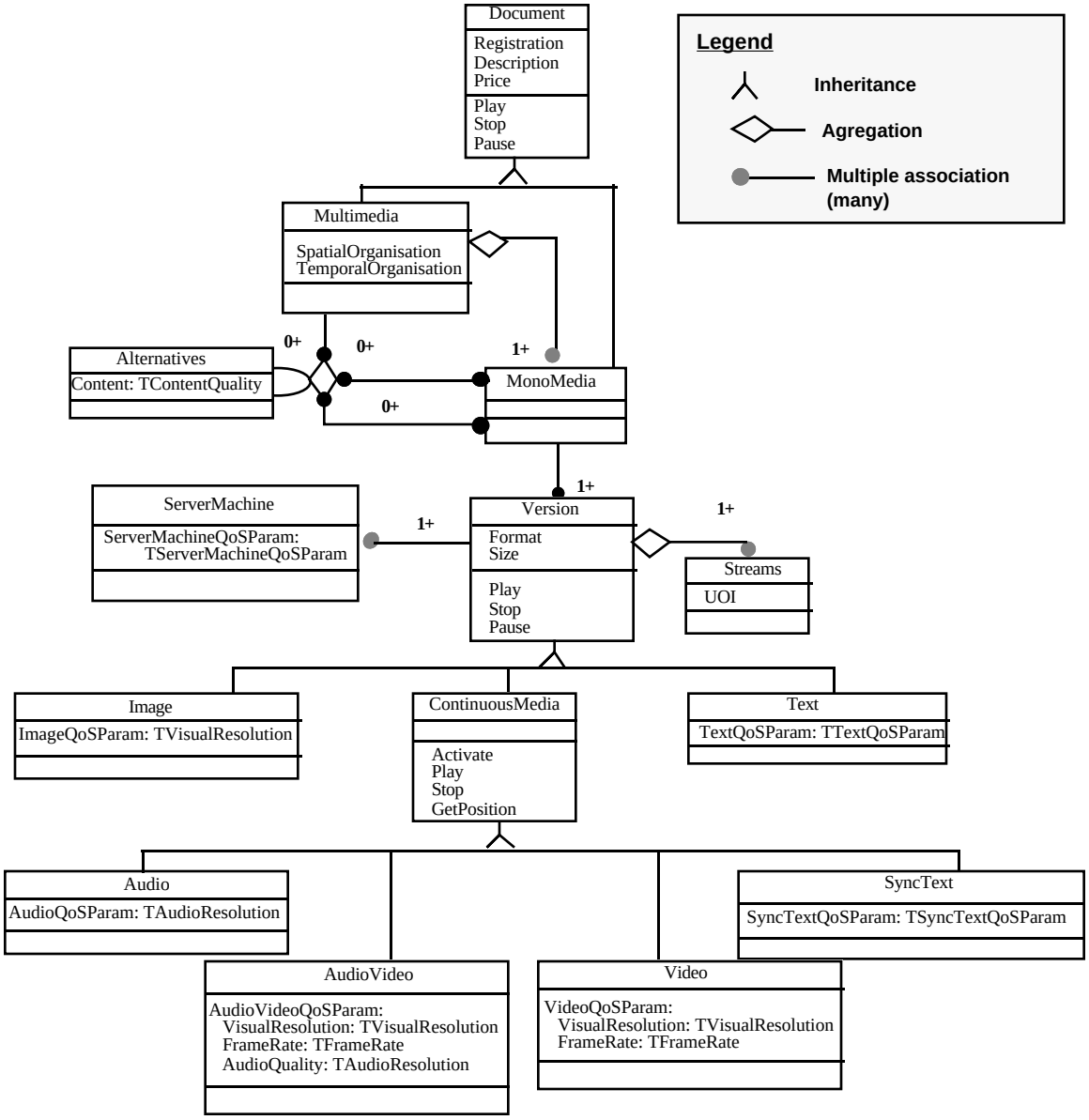


Figure 2: Object Model for Multimedia Documents

A document as shown in Figure 2, is either a multimedia document or a monomedia object. A multimedia document is composed of several monomedia objects usually synchronized with each other and possibly shared by different multimedia documents. In the model presented in Figure 2, the Multimedia entity is described by the aggregation link with MonoMedia documents and by the attributes that depict the spatial and temporal synchronization relationship between the associated monomedia.

These relationships are used during the transfer and display of the document. In addition, a multimedia document includes a Price and information allowing the expression of search conditions on the set of multimedia documents stored in the database. Before displaying a multimedia document, one has to select, for each monomedia object, one of its versions, since we assume that each monomedia may exist in different physical representations, called Versions, which are used by the negotiation protocol. A monomedia object is defined in a particular medium: a text, a still image, an audio sequence, a graphic, a video sequence or an audio-video sequence. Its versions are physical objects represented in the same medium but with different formats and quality. For instance, a monomedia document which is a video sequence may exist in MPEG2 format and also in MJPEG format and under different resolutions.

Each monomedia has different degrees of quality given by its versions. The negotiation must choose one of them according to the desires of the user and the constraints of the client machine. The quality of a given monomedia document is defined by its price, static parameters depending on the kind of monomedia medium or referring to the physical localization of the element. The parameters give, for instance, the format of the coding, the size of the file, the color of a video. They are specific to a version, so they are stored in the version inheritance hierarchy. The parameters related to the machine where the file is located are included in the ServerMachine component of the version. In case of replication of the version, several server machines are needed that have specific quality parameters. In addition, a version of a monomedia object can be decomposed into several streams. The number of streams of a physical monomedia is an additional criteria of quality. Each composition of streams produces a specific Version object of the monomedia with different QoS levels.

The different qualities of a monomedia document are associated to physical monomedia elements -versions- defined on the same medium and holding the same informative content. The negotiation might fail if the quality required by a user for a given monomedia is not available. In this case, the quality of any version doesn't match the requirements. To offer a means to bypass this failure, we propose a second level of negotiation concerning alternatives. This second level affects the informative content of the presentation. The monomedia can be substituted in the context of a given multimedia document by an other monomedia component, called alternative. This object may represent condensed or abstracted information and/or another medium in order to get documents at a lower cost. Such an alternative will be delivered when the QoS offer is not satisfactory for the original monomedia component. For instance, an alternative for a video sequence could be an audio sequence describing the same event or a portion of text describing it.

### 3.2. Profile Manager

In the previous sections, we have seen that the QoS management consists in providing the user with the offer which corresponds to his requirements. The profile manager is in charge of managing the user's QoS preferences; that is, helping the user while setting and modifying his requirements through user profiles. The following operations of the ProfileManager object may be called by an application:

OpenProfileWindow: This operation lets the user select an active profile different from his default profile, and define new profiles for future use.

GetActiveProfile: This operation returns the active profile selected by the user.

NegotiateActiveProfile: This operation is called by the application when the active profile is too restrictive compared to the services that can be provided. Therefore the application will provide as parameters a set of alternative profiles for which services can be provided. The result of this operation should be the selection of one of these profiles or a refusal by the user.

A user profile describes user preferences in terms of (1) QoS settings for video, audio, still images and text, (2) cost he is willing to pay for a given quality, and (3) of time constraints, such as the maximum delivery time. The user QoS setting is described in terms of a set of user-perceived characteristics of the performance of a service. It is expressed in user-understandable language and manifests itself as the set of value for the different QoS parameters.

To avoid repeating the lengthy QoS parameters setting process, the user should be able to store QoS profiles. Then, while starting a new session he selects the desired profile. Furthermore the user can display examples of varying quality in order to see if the profile is pertinent. A set of QoS parameters is associated to each type of monomedia, namely video, audio, text and image. Furthemore to specify the cost and timing constraints a number of parameters are required. The profile manager provides a set of predefined user profiles that help the user in setting a new profile. A detailed presentation of the profile manager can be founded in [5] .

### 3.3. Quality of Service Manager

The purpose of quality of service negotiation is to provide a presentation quality that corresponds to the user's wishes and his financial constraints, as well as within the constraints imposed by the limitations of various system components, such as the available resources at the client's workstation, the bandwidth limitations of the network and the encoding schemes of the available multimedia documents [3] . Therefore this process is based on various management information associated with the various system components. Figure 3 shows the objects within the overall system which are relevant for QoS negotiation. The QoSManager object corresponds to the "QoS" process in Figure 1. The Network, ClientMachine, and ServerMachine (including the continuous media file servers, CMFServer) were also included in that figure. Figure 3 provides some details about the QoS parameters of the Network and ClientMachine that are relevant for the QoS negotiation process.

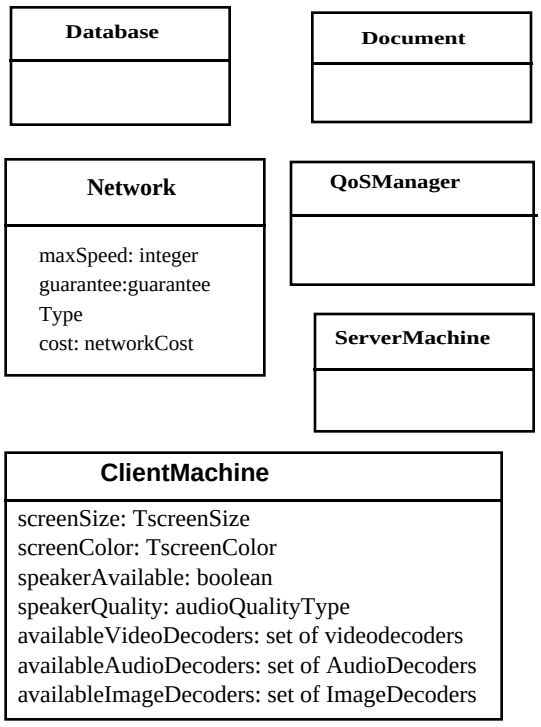


Figure 3: Objects within the Overall System

At the high level of abstraction considered in this section, where distribution is not explicitly taken into account, we associate the management information directly with the objects. For instance, the network has the following QoS attributes: maxSpeed: the maximum throughput of a single connection, guarantee: indicates whether a throughput guarantee can be provided, and cost: a tariff table quoting the cost (per minute) for various throughputs and guarantees. A distinction between static and dynamic management information is sometimes made [3] . The above information is considered static. Dynamic information, such as the available network bandwidth or possible database server congestion, should also be taken into account during QoS negotiation. We assume that such dynamic information is taken into account when the system resources are actually reserved.

The QoS Manager object performs the QoS negotiation and adaptation by interacting with various system components, as shown in Figure 4. This diagram includes the Document and QoSManager objects already shown in Figure 3, and in addition an object representing the application program and additional objects that are involved in the negotiation process. The ProfileManager object is described in Section 3.2, and the objects of type Version are the same as those shown in Figure 2. Figure 4 shows the operations that can be involved on each of the objects. The arrows indicate calling relations between the objects. Two kinds of calls are considered: (a) a standard procedure call by a client object on a server object, denoted by a thin arrow, and (b) the invocation of a "notification" by a server object on a client object (which previously has made a normal call on the server), denoted by a thick arrow. For example, the application may call the negotiatePresentation operation of the QoSManager, whereas the latter may invoke the notification QoSViolation while the user and application are possibly searching through the document using the fastforward, fastbackward and play operations.

A typical scenario of QoS negotiation proceeds as follows. The application may call the operation NegotiatePresentation on the QoSManager, providing as parameter a Document and a userProfile obtained from the ProfileManager. The objective of this operation is to establish, if possible, a configuration of suitable versions of the monomedia in the document that can be presented with a quality of service within the bounds of the given user profile.

If such a configuration is found, then the configuration including the necessary communication links are established, and the document is ready to be played. If such a configuration is not found, the operation returns one or several alternatives of multimedia profiles each corresponding to a possible configuration, which, however, does not conform to the given user profile. The negotiatedPresentation operation may also be called during an ongoing session when a change of the presentation quality or its cost is desired; in this case, the parameter reneg should be set to true in order to indicate that this is a case of renegotiation.

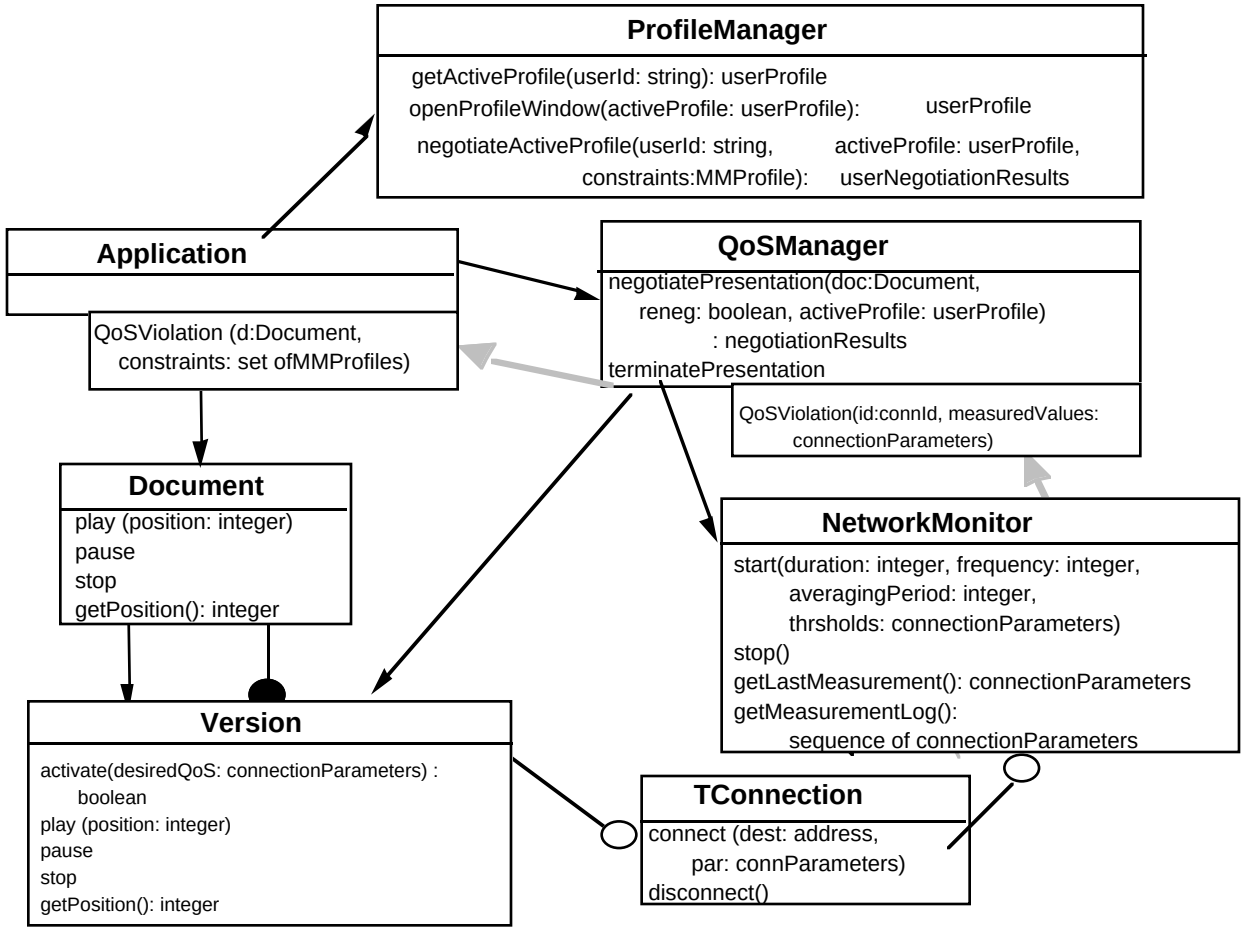


Figure 4: Interactions for QoS Negotiation

### 3.4. Network Monitor

In the case that the network does not guarantee the QoS negotiated during the establishment of a connection, such as in the context of the Internet, it may be useful to monitor the quality actually provided by the network for a given connection. Such monitoring may be useful for switching automatically to an alternative system configuration (if available), without the intervention of the user, when the effective presentation quality does not conform anymore to the profile which was used for the original QoS negotiation.

The object NetworkMonitor shown in Figure 4 performs these monitoring actions. Network monitoring is triggered by the QoSManager by calling the start operation. Parameters of this operation indicate the duration of each measurement period and the frequency with which such measurements should be performed. The NetworkMonitor notifies the QoS manager when the average of the measured values do not satisfy the threshold values provided in the start operation. Measurement values may also be directly obtained by calling operations such as getLastMeasurement or getMeasurementLog.

## 4. QoS Negotiation and Adaptation: Behavior Definitions

After having defined the object types and the operations, the object-oriented analysis and design methods usually lead to the definition of the behavior of the objects involved. Instead of giving a complete definition of the behavior, we will simply explain informally the behavior of the objects described in Section 2, by considering some typical interaction scenarios, which are related to the use cases discussed in section 2. We will complete this section with a discussion of an abstract application programming interface (API).

### 4.1. Establishing a Presentation Session

A typical application program will begin by determining the QoS user profile to be used during the session. For this purpose, it could invoke the GetDefaultProfile operation of the ProfileManager object. As next step, The application will then execute a SearchDocument operation on the Database object with search information obtained from the user.

In order to select a suitable configuration, the application calls the operation NegotiatePresentation of the QoSManager. As further explained in [2] , the QoS manager will consult the meta-data of the document and determine, for each continuous monomedia component, which of the available versions is the best. For the case described in Section 2, the QoS manager will select the version residing on CMFServer 1 for presentation.

For each continuous monomedia component, a real-time transport connection is established over which the coded data can be transmitted from the CMFServer containing the selected version to the client's workstation. The QoSManager will therefore request these connections from the network(s) by calling the Connect operation on a suitable TConnection object. Once a version is associated with an open transport connection, its operation Activate may be called. This operation communicates with the CMFServer and reserves the necessary resources for the real-time data transfer over the associated transport connection. The resources include a synchronization protocol to assure the delivery of data [7] . Once a version has been activated for each monomedia component of the document, the document may be played by invoking the Play operation.

### 4.2. Negotiation with the User

In the case that the QoSManager does not find any satisfying configuration, the NegotiatePresentation operation returns with a failure status and a set of MMProfiles which correspond to certain possible configurations which, however, do not satisfy the active user profile. The application has essentially three choices in such a situation: (a) abandon the presentation of this document, (b) proceed with the presentation according to one of the configurations proposed by the QoSManager, or (c) let the user decide. For realizing choice (b), the application could simply call the Renegotiate operation. In the case of choice (c), the application could call the operation NegotiateActiveProfile on the ProfileManager object and use the resulting UserProfile as the parameter to a subsequent negotiatePresentation operation called on the QoS manager object.

### 4.3. Automatic Adaptation

Let us assume that a presentation session is in progress and that the network becomes congested, thus leading to lower QoS parameters for the transport. Let us assume also that the QoSManager has started network monitoring for this connection with threshold values selected to assure the WorstAcceptedQoS values accepted by the user profile. If the averaged value of the measurements for one of the QoS parameters goes below the threshold, the NetworkMonitor object will notify the QoSManager by calling the QoSNotification operation. The QoSManager has essentially four choices in such a situation: (a) to continue the ongoing session with the present QoS parameters, (b) idem, but also to notify the application of the reduced QoS invoking the notification QoSViolation, (c) to automatically select another configuration, and (d) to call the QoSViolation operation in giving all possible configurations in order to let the application decide.

In the case of automatic reconfiguration (choice (c)), the QoSManager stops the presentation of the document after having obtained the current position by calling the GetPosition operation of the document. It then determines the best alternate configuration, activates this configuration and restarts the presentation by calling the play operation with the position parameter set earlier.

### 4.4. An Abstract Application Programming Interface

An application programming interface (API) of a given software module is usually a set of procedures which can be called by an application program to obtain the services provided by the software module. In the following we describe the functionality of the API provided by the multimedia database and QoS management software and discuss the interfaces at an abstract level.

Given the architecture of our system as explained above, the abstract API is the set of operations that can be called on the various objects within the distributed system. This API should include the following objects and operations:

ProfileManager: operations GetActiveProfile, openProfileWindow, NegotiateActiveProfile

Database: operations Search

Document: operations Play, Stop, Pause, GetPosition

QoSManager: operations negotiatePresentation

Application: the notification QoSViolation could be called by the underlying system components

## 5. Implementation Issues

In this section we discuss some of the issues that come up during the process that leads from an abstract architectural design, as discussed above, to an implementation of the system. In particular, we consider issues related to the distribution aspect of the system and the binding of the abstract API discussed above into interface implementations at the level of the operating system and programming language.

### 5.1. Communication between Distributed Components

In the previous sections, we discussed distribution aspects that are essential for QoS negotiation and adaptation, such as the location of the application in the ClientMachine, the documents on ServerMachines and the presence of networks which provide real-time transport connections for the transfer of continuous

media streams. We discuss in the following some additional issues related to the distribution of the objects introduced earlier.

The database does not reside in the same computer as the application program. Therefore, the implementation of an operation call by the application, such as a call of the search operation, involves a protocol for the remote invocation of the procedure. Various protocols for remote procedure call (RPC) exist, including the OSI standard for "remote operations" (ROSE). One of these protocols could be selected for implementation.

The same problem of remote access arises for the QoSManager which needs to access the QoS information associated with various objects, such as the networks, the database, and the client machine. While the information on the client machine is local, the other objects are remote. While the same RPC protocol mentioned above could be used for this purpose, it may be desirable to use specific protocols developed for distributed systems management, such as SNMP or CMIP, for accessing this management-oriented information. The remote objects in question may already provide access to this information through these protocols independently of our particular application.

A document is actually a distributed object: the meta-data and the non-continuous media is stored in the database server and the continuous media components are stored in continuous media file servers (of type CMFServer). Before preparing a session, the QoS Manager needs to obtain a local copy of at least the meta-data of the pertinent document.

The transmission of (partial) documents involves the complex data structures discussed in Section 3.1 to be coded and transmitted. Some protocol standards have been designed for handling these complex structures, such as SGML, HGML, or MHEG. In our first prototype discussed in Section 5, the document structure following the HyTime standard, was coded in the form of C++ data structures stored in an ObjectStore database. The ObjectStore client software provides access to the server machine using a proprietary protocol for communication.

The Version object introduced in Section 3.1 and used in Section 4 may be considered as an object distributed over the ClientMachine and the CMFServer where its data is located. The first operation that can be invoked is the activate operation. Its function is to establish a connection between the client and the CMFServer through the related transport connection (see Section 4.1). Once this association is activated, the other operations, such as play, may be executed through the collaboration between the client and the server machine. In fact, the execution of the activate operation of a Version object assures that a subsystem structure is established which serves for the real-time processing of the continuous media stream. This subsystem includes, in addition to a protocol for stream delivery and synchronization, allocated system resources for the decoding and playing of the media stream.

### 5.2. API at the Implementation Level

An application programming interface (API), in concrete terms, should be at the implementation level. The abstract API discussed in Section 4.2 was introduced to show the implications of the architectural design on the API that would be provided within the implemented system. Similarly, the standardized application interfaces for specific services, such as graphics systems or communication protocols, are often first defined in abstract interactions that may then be mapped to different concrete interfaces, possibly provided in the context of different programming languages, also called "language bindings" of the abstract interface.

In the following, we give an overview of the concrete interfaces that have been implemented in the News-on-Demand prototype developed within the CITR Major Project on "Broadband Services". This prototype realizes most of the functions described in this paper (the automatic adaptation is only foreseen for March 1996), although it has not been developed based on the architecture described here. However, we think that its concrete interfaces may be easily put in correspondence with the abstract interfaces described in this paper.

The implementation language of the prototype is C for the objects Version and TConnection, and C++ for the other objects. Using the distributed ObjectStore database software, the access procedures to the database become local procedure calls. Therefore the interfaces for the Database, the ProfileManager, the QoSManager and for access to the meta-data and non-continuous monomedia components of documents are simply programmed as C++ procedures. The interface with the NetworkMonitor is also programmed as C++ procedures, however, their execution involves inter-process communication for invoking the corresponding operation on the NetworkMonitor object (supervisory part) which is implemented as a separate process.

The local interfaces to the Version and TConnection objects is implemented in the form of C procedures. Since C does not directly support the concept of object instances, a single software module, representing a Transport entity, supports a large number of transport connections, identified by a local connection identifier. The procedures corresponding to the operations of the TConnection object have one additional parameter which is the connection identifier of the connection. A similar approach is taken for the interface with the Versions.

## 6. Conclusion

We have presented a new architectural design of the adaptive distributed multimedia system we have developed for remote access to multimedia databases. The system is based on a quality of service negotiation

and adaptation protocol allowing the selection of the best configuration for a given user request and for automatic adaptation in case of changes to the system parameters. We have introduced a powerful notion of user profile to keep the user preferences. To describe the functional aspect of the system, we have detailed the collaborations between its main actors. Realistic scenarios are given to illustrate the system's behavior. Four managers compose the system: the database manager, the profile manager, the QoS manager and the network monitor. We have described the high-level system structure with an object-oriented approach and defined the semantic of the different components by explaining their role within the context of QoS negotiation and adaptation. Finally an abstract application programming interface was given which is believed to be suitable for various multimedia applications, and can be refined into the actual API used at the implementation level.

This work is done in the framework of a collaborative research project funded by the Canadian Institute for Telecommunication Research. This project resulted in an integrated prototype that was demonstrated in August 1995. The experience gained in the development of this first prototype of the NewsOnDemand system leads us to describe a high-level specification of a general adaptive distributed multimedia system. A new version of the prototype is currently being designed and developed. We think that the architectural design we have presented here will be helpful for the evolution of the system.

**Acknowledgments**

This work grew out of the discussions about the architecture of the NewsOnDemand prototype developed within the CITR "Broadband Services" project. We thank all the participants in this project for fruitful discussions. Special thanks are due to Rolf Velthuys for his major contribution to the architecture of the prototype.

**References**

[1] Dubois, E., Baaziz, N., & Matta, M. (1995). *Impact of Scan Conversion Method on the Performance of Scalable Video Encoding*. In IS&T-SPIE, San Jose, USA:

[2] Hafid, A., Bochmann, G. v., & Dssouli, R. (1995). *Models for QoS negotiation in distributed multimedia applications*. In Second International Workshop on Protocols for Distributed Systems, Salzburg, Austria:

[3] Hafid, A., Bochmann, G. v., Kerhervé, B., Dssouli, R., & Gecsei, J. (1995). *On Quality of Service Negotiation for Distributed Multimedia Applications* (Technical Report No. 977). Université de Montréal, Canada.

[4] Hutchinson, D., Coulson, G., Campbell, A., & Blair, G. (1994). *Quality of Service Management in Distributed Systems*. In M. Sloman (Eds.), Network and Distributed Systems Management (pp. 273-303). Addison-wesley.

[5] Isnard, B. (1995). *Gestion des profils de qualité de service* (Internal Report). Université de Montréal, Canada.

[6] Kerhervé, B., Vogel, A., Bochmann, G. v., Dssouli, R., Gecsei, J., & Hafid, A. (1994). *On Distributed Multimedia Applications: Functional and Computational Architecture and QoS Negociation.* In M. I. G. Neufeld (Ed.), IFIP Fourth International Workshop on Protocols for High-Speed Networks, (pp. 21-37). Vancouver, Canada: Chapman & Hall.

[7] Lamont, L., & Georganas, N. (1994). *Synchronisation Architecture and Protocols for a Multimdia News Service Application*. In IEEE Multimedia Computing and Systems. Boston:

[8] Nahrsted, K., & Smith, J. (1995). *QoS Broker*. IEEE Journal of Multimedia Systems, vol 2 no 1 (spring 1995) , 53-67.

[9] Neufeld, G., Makaroff, D., & Hutchinson, N. (1994). *The design of a file server for scalable VBR media* (Technical Report No. University of British Columbia, Vancouver, Canada.

[10] Özsu, T., Szafron, D., El-Medani, G., & Vittal, C. (1995). *An Object-oriented Multimedia Database System for a News-on-demand Application*. ACM Multimedia Systems, 3 (November 1995) , 182-203.

[11] Parnas, D. L., & Clements, P. C. (1986). *A rational design process: How and why to fake it.* IEEE Transactions on SE, SE-12 (2) , pp. 251-257.

[12] Rambaugh, J., & al. (1991). *Object-oriented Modeling and Design*. Prentice Hall.

[13] Velthuys, R., & al, a. (1995). *CITR Broadband Services: the March'95 demo* (Technical Report). University of Waterloo, Canada.

[14] Vogel, A., Kerhervé, B., Bochmann, G. v., & Gecsei, J. (1995). *Quality of Service Management: a survey*. IEEE Journal of Multimedia Systems, Vol 2 no 2 (Summer 1995) , 10-19.