Enabling Technology for Distributed Multimedia Applications.

Article *in* Ibm Systems Journal · January 1997

DOI: 10.1147/sj.364.0489 · Source: DBLP

Gregor V. Bochmann
University of Ottawa

Eric Dubois
University of Ottawa

M. Tamer Özsu
University of Waterloo

Abdelhakim Senhaji Hafid
Université de Montréal

Enabling Technology for Distributed Multimedia Applications

J.W. Wong, K.A. Lyons, D. Evans, R.J. Velthuys,

G.v. Bochmann, E. Dubois, N.D. Georganas, G. Neufeld, M.T. Özsu,

J. Brinskelle, A. Hafid, N. Hutchinson, P. Iglinski, B. Kerhervé,

L. Lamont, D. Makaroff, and D. Szafron

May 12, 1997

Abstract

In September 1993, the Canadian Institute for Telecommunications Research, in collaboration with the IBM Toronto Laboratory Centre for Advanced Studies, initiated a major project on Broadband Services. The goal of this major project is to provide the software technologies required for the development of distributed multimedia applications. Of particular interest are "presentational" applications where multimedia documents, stored in database servers, are retrieved by remote users over a broadband network. Emphasis is placed on efficiency and service flexibility. By efficiency, we mean the ability to support many users and many multimedia documents. As to service flexibility, we mean the application is able to support a wide range of quality of service requirements from the users, adapt to changing network conditions, and support multiple document types. The research program consists of six constituent projects: multimedia data management, continuous media file server, quality of service negotiation and adaptation, scalable video encoding, synchronization of multimedia data, and project integration. These projects are investigated by a multi-disciplinary team from eight institutions across Canada. Multimedia news has been selected as a target application for development, and the results from the various projects have been integrated into a multimedia news prototype. In this paper, the system architecture, research results, and the prototyping effort, are presented.

1 Introduction

In recent years, advances in computer and networking technologies have led to the development of powerful workstations with audio and video capabilities, server machines with high capacity storage devices, and broadband networks that support quality of services (QoS) guarantees. These advances have spurred interest in the development of distributed multimedia applications. Deployment of such applications would be facilitated by the availability of service enabling software that hides the details of the underlying network infrastructure from the application developer. Research is also required to fully understand the communication requirements of these applications and the corresponding implications for system and network design.

An important class of distributed multimedia applications is "presentational" applications where multimedia documents featuring continuous (audio and video) and/or discrete (image and text) data are accessed interactively by remote users. Application areas include multimedia news, digital libraries, home shopping, and distance education. The success of this type of interactive service is heavily dependent on the ability to deliver the service to a large community of users in an effective manner.

In September 1993, the Canadian Institute for Telecommunications Research (CITR), in collaboration with the IBM Toronto Laboratory Centre for Advanced Studies (CAS), initiated a major project on Broadband Services. The goal of this major project is to provide the software technologies required to support the development of distributed multimedia applications. Our work is focused on presentational applications, where emphasis is placed on efficiency and service flexibility. By efficiency, we mean the ability to support many users and many multimedia documents. As to service flexibility, we mean that the application is able to support a wide range of QoS requirements from the users, adapt to changing network conditions, and support multiple document types. The research program is organized as six constituent projects, which are investigated by a multi-disciplinary team from eight institutions across Canada. An important activity is the development of an integrated prototype using the research results from the constituent projects. Multimedia news has been selected as a target application for development — collaborators at IBM Canada played a major role in this selection. In this paper, the system architecture, research results, and prototyping effort are presented.

This paper is organized as follows. In Section 2, we discuss the key design decisions and the organization of the Broadband Services research program. In Sections 3, the details of the overall system are described. These includes the application programming interface, the various software modules, and the communications among the various modules. Our effort in developing an integrated prototype, and the challenges that we have encountered, are also discussed. Besides the integrated prototype, the research team has made advancements to the various research areas under investigation. These accomplishments are discussed in Section 4. Finally, Section 5 contains some concluding remarks and a discussion of future research directions.

2 Design Decisions and Project Organization

2.1 Design Decisions

Conceptually, our system is a distributed system where multimedia documents, stored in databases and file servers, are accessed by remote users over a broadband network. Its design is based on the following decisions:

1. **Uniform Treatment of Content Data and Meta-Data:** In a multimedia document, the *content data* correspond to the text, image, audio, and video data, and *meta-data* contain descriptive information about the content data. Meta-data include annotation information such as keywords, author, and date of creation, as well as information which is relevant to system operation such as document structure and encoding schemes for audio or video. Our system treats meta-data and content data uniformly from the perspective of querying.

2. **Use of an Object-Oriented Database:** In general, a multimedia document is a structured complex object containing a number of media objects (video, audio, image, or text). Video

and audio objects are generally large, consisting of digitized samples of analog data. There is no simple structure to these objects as there is to, for example, the name, address, and salary attributes of an employee object in a traditional database management system (DBMS). Video and audio objects also have temporal and spatial relationships to one another. Relational DBMSs are not suitable for supporting multimedia documents because (i) they are designed to efficiently manage large numbers of small objects, and (ii) they manage fixed data types and are not extensible. An object-oriented DBMS is adopted because features such as abstraction and encapsulation of complex objects, an extensible type system, and support for representing various hierarchies, are more suitable to meeting the requirements of multimedia applications.

3. **Development of a Continuous Media File Server:** Content data come in two varieties depending on whether the data are continuous (audio and video) or discrete (image and text).[1] Image and text data are stored in the DBMS. For audio and video data, our system must provide guarantees of delivery, as well as support for synchronization of independent media streams and QoS. We have therefore decided to develop a special purpose continuous media file server. A consequence of this decision is that continuous data and discrete data may be stored on separate servers.

4. **Synchronization of Multiple Media Streams:** In our system, the media objects that make up a multimedia document may be stored on different media servers. This facilitates the development of applications where the same video stream may be combined with one of several possible audio streams such as those corresponding to different languages. A mechanism is needed to request the delivery of media objects from different servers and to synchronize their presentation at the client.

5. **QoS Negotiation and Adaptation as an Integral Part of the System Architecture:** To achieve service flexibility, an application must be able to cope with varying network con-

[1] In some cases, image and text data may be considered as continuous. For example, closed captioning involves text that must be displayed in a timely manner.

ditions as well as varying presentation quality requested by the users. The latter is relevant, for example, when a video document is available at multiple levels of resolution. The QoS negotiation process is guided by the user's preferences and priorities which can be captured in the form of user profiles. The system also adapts to changing user priorities, system parameters, and resource availability. In our system, the various system components must effectively support QoS negotiation and adaptation.

2.2 Project Organization

The CITR Broadband Services research program consists of six constituent projects: multimedia data management, continuous media file server, quality of service negotiation and adaptation, scalable video encoding, synchronization of multimedia data, and project integration. These projects are organized around different system components, and are investigated by a multi-disciplinary team from eight institutions across Canada. The overall major project is led by J. Wong of the University of Waterloo. The co-leader is K. Lyons of the IBM Toronto Laboratory Centre for Advanced Studies. Apart from project management, an important responsibility of the major project leader and co-leader is to coordinate the milestones of the constituent projects so that the objectives of the major project are met.

3 System Description

Our system architecture, developed as a result of our design decisions, is depicted in Figure 1. It is based on a client-server model. In this section, we first describe the features of our multimedia news application, and then the software technologies that can be used to develop such an application. These include the various software modules, and the communications between modules. For ease of exposition, we will base our description on an example document and selected user requests.

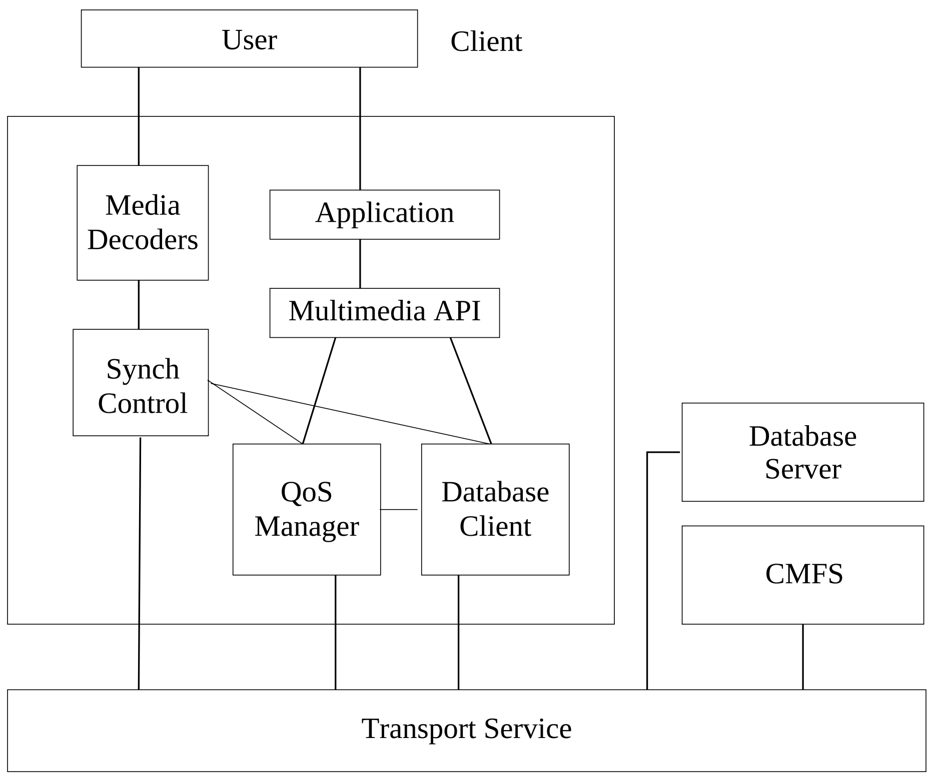

Figure 1: System Architecture

3.1 Multimedia News Application

Our multimedia news application is designed to support the search, retrieval, and presentation of multimedia news documents. In general, a news document may contain media objects such as audio, video, image, and text. These objects may be stored on different servers. To present a document, the corresponding media objects are retrieved and synchronized for presentation at the client workstation. Facilities are provided for a user to negotiate the quality of the presentation with the system such as encoding scheme and frame rate of video. Facilities are also provided for users to store their QoS preferences in user profiles.

3.2 Multimedia API

A multimedia API has been defined to support application development. Table 1 contains an overview of the API primitives, organized by function groups [21]. This is a minimal API, which has been implemented as a C++ class library [21], using the capabilities provided by the software modules developed by the constituent projects. Details of this implementation are provided in Section 3.4.1.

Our multimedia news application has been developed using this API.

3.3 Document Representation

We next discuss our approaches to document representation and document storage. This will facilitate our discussion of the various software modules in later sections.

3.3.1 SGML/HyTime Standard

We follow the SGML/HyTime standard [1, 2] for representing document structure. SGML formally specifies document structure by defining element types such as **paragraph** and **figure** and the relationships between them using a document type definition (DTD). SGML does not prespecify the nature of these elements, nor the structure of the composition hierarchy that contains them. Instead, a document designer specifies a different DTD for each category of document being designed.

Function Group	Primitive	Explanation (where required)
Initialization	initialization log-on	acquire resources, and establish connections between system components
Search	set search scope search on meta-data search on content	set range of documents to be searched
Retrieval and document presentation	prepare	prepare for retrieval; this includes the pre-fetching of data
	present	start or resume presentation
	pause	
	fast forward/rewind	
	close	terminate presentation
QoS negotiation	open profile window get active profile negotiate active profile get document QoS get system QoS negotiate presentation	
Shutdown	log-off	
	release	release resources and terminate connections

Table 1: Multimedia API

The representation of spatial and temporal relationships between media objects is an important consideration because such information is required to plan the retrieval and presentation of multimedia documents. In following the HyTime philosophy, we completely separate the presentation of a document from its content. This has two implications. First, the user's presentation style preferences must be stored and accessed when necessary. The second and arguably more important consideration is to represent the spatial and temporal relationships in accordance with the HyTime standard.

HyTime defines a number of architectural forms; one of them is the finite coordinate space (FCS) which is used to model spatial and temporal relationships. We define an FCS of three dimensions: x and y to represent spatial dimensions and *time* to model the temporal dimension. A set of ranges along these axes forms an *extent* which corresponds to an *event* [1]. An event schedule is used to represent temporal relationships among the various media objects. Within this context, our model of spatial and temporal relationships is a set of type definitions that correspond to the relevant HyTime concepts. Details of the type system design can be found in [23, 24].

3.3.2 Example Document

We show in Appendix A a complete SGML markup of a multimedia news document that describes the CITR Broadband Services demo at the CASCON '96 conference in Toronto. This document, as viewed by a user of our prototype, has the layout shown in Figure 2.

The markup begins with a standard SGML **doctype** tag (or element) identifying the DTD to which the document conforms. This is followed by the **article** element, which identifies the document's unique ID and its primary language. The **frontmatter** element contains sub-elements representing the meta-data used for document indexing and searching. It includes the document's editorial information (author, date, keywords, location, subject, and source), the document's headlines and abstract.

The next section of the document is marked by the **async** element. It contains the asynchronous content which may be displayed at any time and with no timing constraints; text and images fall

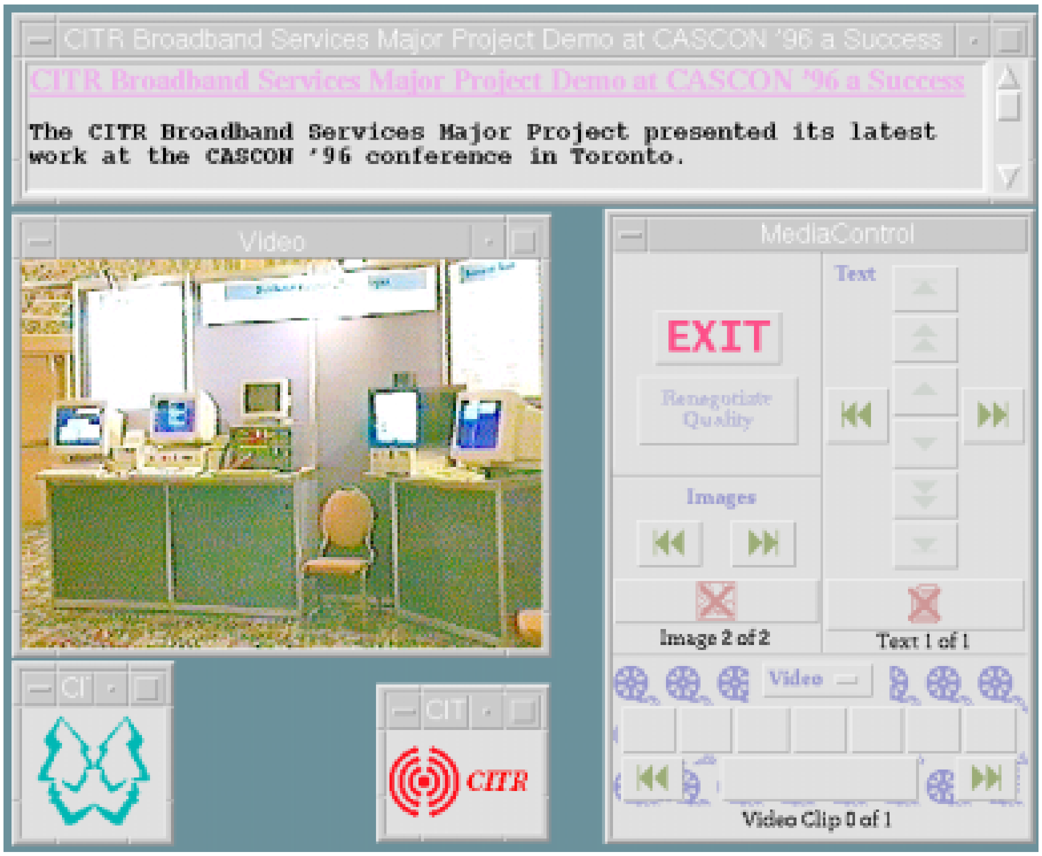

Figure 2: Sample document as presented by the application

into this category. We have two **image** elements corresponding to the two logos in Figure 2. Each **image** element defines an image but makes no mention of its content; the content is described using a separate **image-variant** element. In general, an **image-variant** element includes an ID, the name of a file containing the image data, the size of the image data, and QoS parameters such as encoding scheme used, the image's colour space, and the image's dimensions. Note that multiple variants of the same image may be used to provide multiple QoS levels. Our example document, however, contains only one variant of each image.

The next part of the document is marked by the **sync** element, and contains specifications for the document's continuous media objects such as audio and video. There are four components to this specification: axis definition, object declaration, object extent lists, and QoS variants. The SGML elements **x**, **y**, and **time** define the HyTime axes which will be used to place and measure media objects. These objects are then declared using **audio** and **video** elements, nested within an *event schedule* (**av-evsched**) and a finite coordinate space (**av-fcs**). The object declarations include the price of retrieving the object and a list of QoS variants that represent the object's content.

The object extent lists define, in the **av-extlist** element, the sizes and positions of objects that have been defined previously. This allows different objects to occupy different spaces within the FCS. An extent list is composed of (start, size) pairs, one for each of the HyTime axes previously defined. This information is used in media stream synchronization. Finally, the details of the QoS variants for each object are specified. Our example document contains two variants for each of the audio and video objects. The various user-level QoS parameters are included with the **audio-variant** and **video-variant** elements. These parameters are used in QoS negotiation. Each QoS variant refers to a **stream** element, which is the link between the database and the continuous media file server (CMFS). Audio and video objects stored in our CMFS are identified by universal object identifiers (UOIs). Each **stream** element specifies a UOI and the size of the object.

When our example SGML document is parsed, it is converted into objects suitable for insertion into the database. The text components of the document (i.e., title and abstract) are included in these objects. The two image files are read in and used to provide the content for the image objects in the database.

3.4 Software Modules

In our design, applications reside above an API module, which communicates with the other software modules in the system (see Figure 1). These modules communicate with each other in order to achieve the desired results. In this section, an overview of each software module is presented.

3.4.1 API module

The API module insulates the application developer from the workings of the rest of the system. It provides a set of objects which enable the developer to use the entire functionality of the system without being aware of the components involved or how they interconnect. These objects are:

- **Session Manager:** This object represents an application's connection to the rest of the system. It should be the first item constructed by the application; likewise its destruction should be the final operation performed before the application terminates.
- **Query Manager:** This object is the application's interface to the database management system (DBMS). It manages the construction of queries, performs the actual query operations, and makes the results available.
- **Document Manager:** This object is associated with an entire multimedia document. The methods within this object allow for the retrieval and examination of the document's content. There is a separate document manager for each document currently being processed.
- **Presentation Managers:** These objects are associated with the various media components of a document. For a given media component, the corresponding presentation manager has knowledge on how to present the data to the user. In addition, presentation managers that represent continuous media data, e.g., the audio and video, also implement a "virtual VCR" interface, which allows the application to control the presentation, to query the current temporal position, and to query the current state of the presentation (stopped, playing, etc.)

3.4.2 DBMS module

The database is an object-oriented system that complies with the SGML/HyTime standard. It is based on ObjectStore** [16] and therefore the ObjectStore API is used to search and retrieve documents from the database. Facilities are provided to retrieve a set of documents, to iterate through the set, and to retrieve a document based on the instance variables of its objects. ObjectStore is organized with a server component and a client component; these components are shown as database server and database client respectively in Fig 1. For convenience, we will use database to mean the two components working together.

3.4.3 QoS Manager module

The QoS manager module is responsible for managing user profiles, validating presentation requests through these profiles, and performing QoS negotiation.

3.4.4 Synchronization module

The synchronization module is responsible for the presentation of media objects. Once the QoS manager has determined the QoS variants that should be used for the presentation, it passes this information to the synchronization module, which then manages the synchronized presentation.

3.4.5 The CMFS module

The CMFS module provides a set capability to deliver continuous media data to the client. It also provides support for QoS negotiation and synchronization. Included in this module is the transport of continuous media data over an ATM network.

3.5 Communication between software modules

We will illustrate the communication between the different software modules by means of two examples: (i) processing of a query and (ii) presentation of a document.

**Product names that may be marked by a double asterisk may be trademarks of their respective owners.

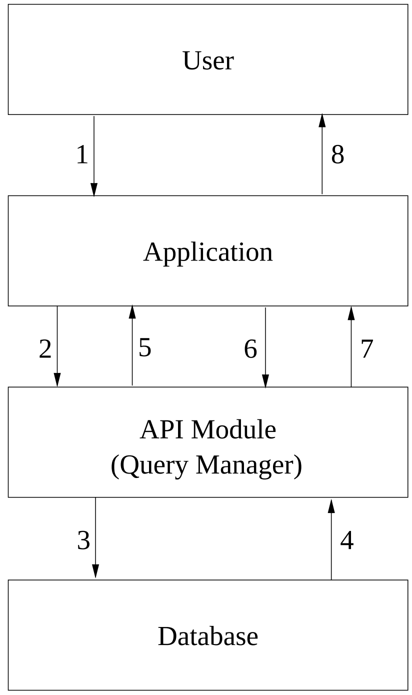

Figure 3: Steps required to process a query

3.5.1 Processing of a Query

The software modules involved in processing a query are depicted in Figure 3. In this figure, communication between software modules are shown by the steps indicated. These steps are explained below.

1. The user requests a keyword search (e.g., find all documents that include reference to CITR and CASCON).

2. The application collects the user's search criteria and passes them to the query manager in the API module; the query manager looks after the retrieval of documents.

3. The query manager first retrieves the ObjectStore database root corresponding to all documents and then retrieves the meta-data of each document.

4. Upon receiving a response from ObjectStore, the query manager checks each document's meta-data to see if the attributes match the user's request. Suppose there are more than one matching documents. The objects representing these documents are converted into objects that are understood by the application and the API.

5. The query manager returns to the application the status of the query (which is success in this case).

6. The application requests information about the matching documents.

7. The query manager responds with the meta-data of the matching documents.

8. The application presents the user with these matching documents.

3.5.2 Presentation of a Document

In our second example, the user wishes to view a document that matched the search criteria. The software modules involved and the communications between modules are shown in Figure 4. The steps are explained below.

1. The user requests the retrieval and presentation of a specific document.

2. The application contacts the document manager corresponding to the requested document and asks it to retrieve and present the document.[2]

3. The document manager requests the presentation managers associated with the document to retrieve and display their respective media objects.[3]

   In the following description, steps 4 to 6 are performed for each presentation manager that corresponds to a discrete media object, while steps 7 to 16 are performed for each presentation manager that corresponds to a continuous media object.

[2] In our system, there is a document manager for each document that matches the search criteria.

[3] There is one presentation manager for each media object.

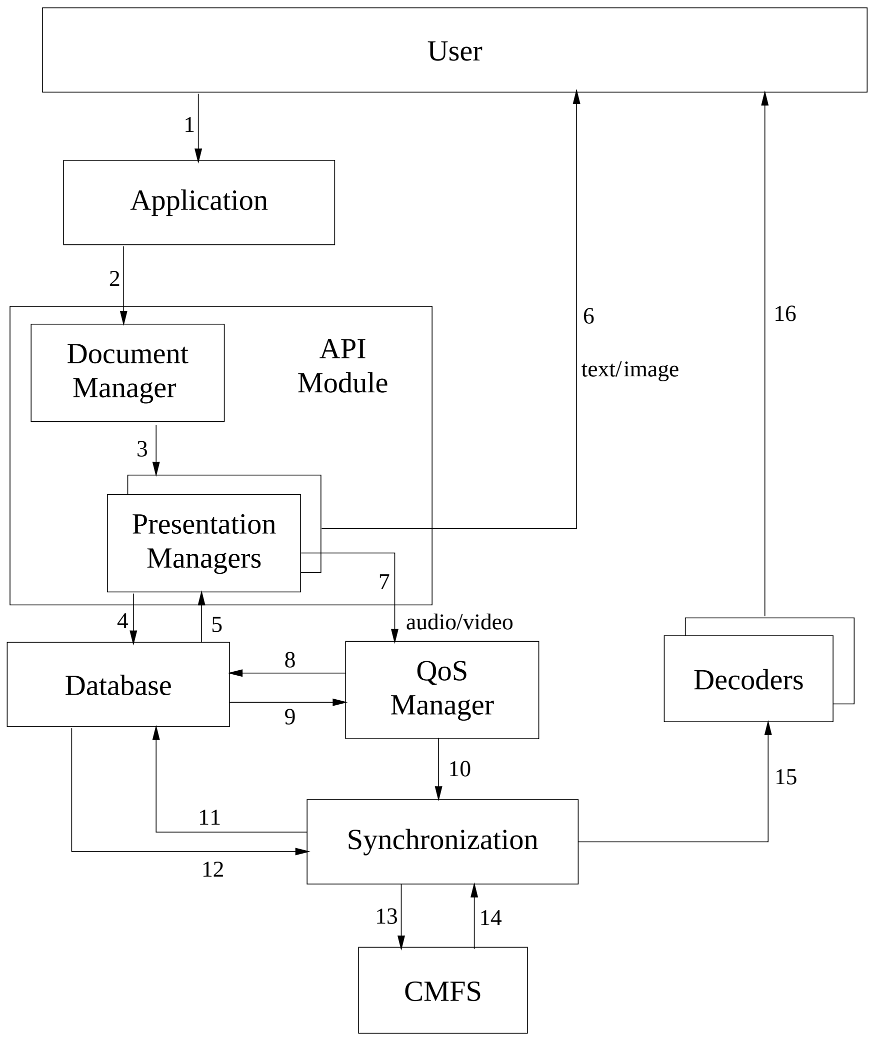

Figure 4: Steps required to display a document

4. The presentation manager retrieves its associated media object from the database.
5. The database returns with the media object.
6. The presentation manager displays the contents of the media object on the screen.
7. The presentation manager requests the QoS manager to negotiate QoS parameters and to display its associated continuous media object.
8. The QoS manager queries the database for the QoS parameters of each available media variants for the requested document.[4]
9. The database returns the QoS variant information; the QoS manager then selects the variants that are appropriate to the user's profile.
10. The selected variants are passed to the synchronization module for display.
11. The synchronization module queries the database for additional information about the selected variants, such as formats, HyTime parameters, UOIs, etc.
12. The database responds with the requested information. This information is then used to construct the presentation scenario, to configure the media decoders and to access the CMFS.
13. The synchronization module requests the CMFS to deliver the media objects.
14. The CMFS sets up the required connections, and sends data packets continuously to the synchronization module.
15. As frames of media are received, the synchronization module determines when they should be displayed, and at that time passes them to the appropriate media decoders.
16. The media decoders display the frames to the user.

[4] The current implementation does not include variants for image and text.

3.6 Prototype Development and Integration Challenges

We have been successful in meeting our milestones and delivering versions of our integrated prototype. Much of the success is due to the effort of the project integration team, which was led by R.J. Velthuys (September 1994 to June 1996) and by D. Evans (since July 1996). This team is comprised of research staff and graduate students from all participating institutions. Team members collaborate at both the design and implementation levels. The design level is concerned with the definition of the reference architecture. Of particular importance are the interfaces between system modules. At the implementation level, integration team members interact frequently by email, phone calls, and short term visits. Much progress was made at an integration workshop organized by K. Lyons of IBM CAS in November 1995. Members of the integration team spent two weeks at IBM CAS, working out the details of the interfaces, modifying the software modules as required, collaborating in debugging these modules, and producing an enhanced version of the integrated prototype.

The latest version of the integrated prototype was demonstrated at the CASCON '96 conference in November 1996. It has the following features. The ObjectStore server and the CMFS reside on an IBM RS/6000* running AIX*. Two client platforms are supported: IBM RS/6000 running AIX and SUN SPARC** running SunOS**. The networking technology is based on ATM switching equipment from Newbridge Networks**, and audio/video support is provided by motion JPEG cards from IBM and Parallax**. The multimedia news application contains a news browser with facilities for QoS negotiation, searching, retrieving, and presenting news documents.

While developing our prototype, we have gained much experience in working together as a team. We have also encountered challenges that need to be addressed. These challenges are discussed below.

- The CITR Broadband Services major project involves researchers from eight institutions across Canada. Very often, research teams at the different institutions progress at different speeds,

*AIX, IBM, and RS/6000 are trademarks of IBM Corporation.

**Product names that may be marked by a double asterisk may be trademarks of their respective owners.

resulting in the need to develop simplified versions of some software modules for testing purposes, while waiting for these modules to become ready.

- Despite the effort of the project integration team, there were still instances where the definition of interface primitives was misunderstood. Much effort was spent in clarifying these primitives.

- The computing environment at the different institutions is not identical, e.g., operating system and compiler. This has led to added effort in understanding the differences in system configuration, before we can successfully install the same software modules at different institutions.

- We have been using leading edge equipment in our prototype development. This includes ATM switches, ATM adapter cards, and motion JPEG cards. Much effort was required to understand how this equipment works. Compromise was made because of the capability of this equipment. For example, we were planning to develop a transport service over a native ATM interface, but such an interface was not available. The compromise was then to run UDP over IP over ATM.

- Much effort was spent in the first year of the major project to bring the team members together to work as a team. This is due to the differences in established culture at the different institutions and the conflicting priorities among researchers at different institutions.

- During our prototype development, we have to make compromises on the features to be included because of resource limitation. For example, in our synchronization module, much effort was spent on the design and implementation of algorithms to synchronize audio and video objects; work on the temporal synchronization of audio/video with image/text has been delayed. As another example, we have been successful in the design and implementation of algorithms for scalable video encoding and decoding. Such algorithms are attractive when one wishes to effectively support video at different levels of resolution. Unfortunately, our software implementation is very slow, and hardware decoding devices are not available. We have therefore decided not to include the scalable video feature in our integrated prototype.

4 Research Accomplishments

Apart from the collaborative effort that has led to the development of our integrated prototype, constituent projects have made advancement to the state of knowledge in their respective areas of research. In this section, these research accomplishments are discussed.

4.1 Multimedia Data Management

As mentioned previously, our database management system (DBMS) is based on ObjectStore. To provide support for multimedia and be compliant with SGML/HyTime, we have built an extension layer on top of ObjectStore.[5] This extension layer consists of the type system, and addresses three issues related to multimedia data management: modeling of the basic media components (i.e., text, image, audio, and video), document representation based on SGML/HyTime, and the capture and storage of meta-data. An example of a SGML marked-up document has been provided in Appendix A. This document is consistent with the document type definition (DTD) for a news document. In general, we should be able to store different types of documents in one database by accommodating multiple DTDs. Tools should also be available to automatically insert marked-up documents into the database.

4.1.1 Dynamic Insertion of New DTDs

Our system is designed to handle multiple DTDs and support the creation of types that are induced by these DTDs. It analyzes new DTDs and automatically generates the types that correspond to the elements they define. We store the DTD as an object in the database so that users can run queries like "Find all DTDs in which a **paragraph** element is defined." (see Figure 5).

In our design, a meta-DTD describes a grammar for defining DTDs, and a DTD parser parses each DTD according to this grammar. While parsing the DTD, an object is created for each valid SGML element defined. This object contains information about the element, such as its name, attribute

[5] Our system is based on ObjectStore version 4, which does not provide support for multimedia objects. Support for multimedia, e.g., media managers, is provided in the recently released version 5.

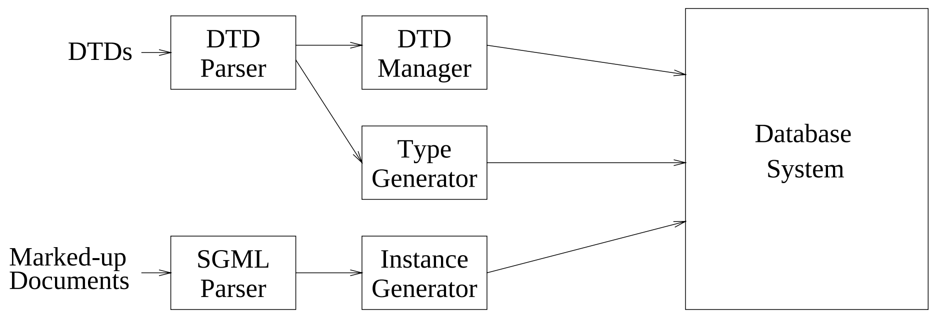

Figure 5: DTDs and Document Entry Tools

list and context model. If the DTD is valid, a type generator is used to automatically generate C++ code that defines a new ObjectStore type for each element in the DTD. For example, if a `Book` DTD is parsed, objects representing `Title`, `AuthorList`, `Chapter`, `Section`, `Paragraph`, `Index`, etc., would be created. We have also addressed two important problems related to abstraction so as to reduce the complexity of the multimedia type system and therefore reduce maintenance time and errors. First, if two or more elements in the same DTD share a common feature, then that feature is automatically extracted and promoted to an abstract superclass. For example, the `Video` and `Audio` types both share a common duration attribute, so the abstract supertype `Temporal` was created to promote this feature. The second problem is related to common element definitions across different DTDs. In general, this is a difficult problem because it leads to the well-known semantic heterogeneity problem that has been studied extensively within the multi-database community. It involves the ability to determine whether two elements are semantically equivalent. In our design, we have chosen to give up some abstraction in favor of a semantically "safe" type system. Specifically, we only re-use types that have well-defined semantics, e.g., atomic types such as `Image` and `Text` and high-level abstract supertypes such as `TextElement`, `Structured`, and `HyElement`. For the rest of the elements in a given DTD, we create new types. Name conflicts between elements in different DTDs are resolved automatically by using the DTD name as prefix during type creation (for example, `article_section` and `book_section`).

A major advantage of our approach is that new element types are inserted into the database without costly schema evolution. The DTD manager takes the DTD file as input and stores the DTD as an object in the database that can later be used for parsing documents. As soon as a DTD is stored in the database, SGML documents of that type can be inserted. Further details of handling multiple DTDs can be found in [25].

4.1.2 Automatic Insertion of Multimedia Documents

Tools for insertion of documents into a database are not developed in many multimedia DBMS projects because they are considered to be outside the scope of database work. However, such tools have been developed for our systems. Our approach is to couple the database with a retrofitted SGML parser[6] [6]. This parser accepts an SGML document instance from an authoring tool (see Figure 5). It then fetches the required DTD from the database and uses this DTD to validate the document instance. If the document is error-free, an output is generated and passed to an instance generator (see Figure 5. This output is in the form of a parse tree, and includes a text string for the document that is stripped of the markup, together with a linked list of nodes containing annotations into the string, an attribute list, and pointers to parent and next nodes. The instance generator traverses the parse tree and instantiates the appropriate objects in the database corresponding to the elements in the document. These are persistent objects, and can be accessed using the query interface.

4.2 Continuous Media File Server

We have designed and implemented a high-performance continuous media file server (CMFS) that is scalable, and provides support for QoS and synchronization.

[6] This parser is based on a freeware application called `nsgmls` developed by James Clark, available from ftp://ftp.jclark.com/pub/sp/.

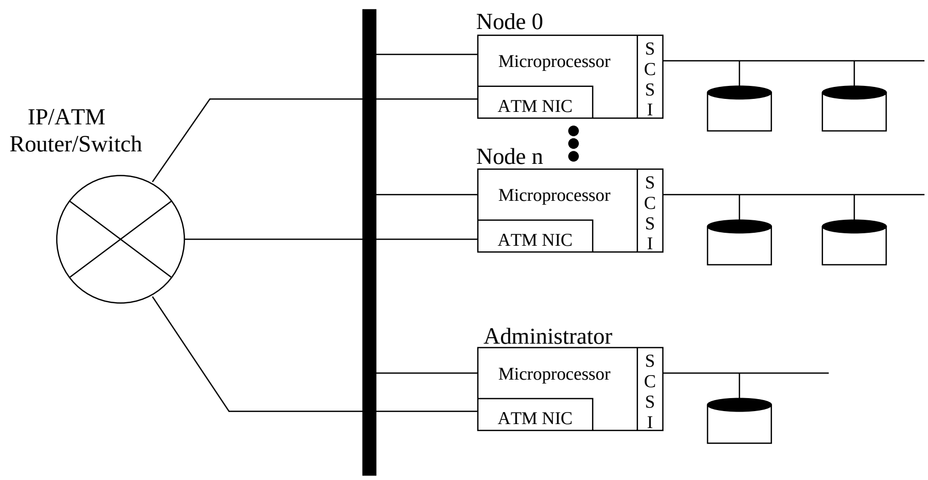

Figure 6: Continuous Media File Server Architecture

4.2.1 CMFS Architecture

Our CMFS is based on a set of server nodes, each with a processor and disk storage on multiple local SCSI-2 Fast/Wide buses. One of these nodes is the administrator node. To achieve high-speed operation, the server nodes are connected to an ATM network (see Figure 6). A sufficient number of disk drives are attached to the SCSI buses to provide the required bandwidth. The disks can be striped along a single SCSI bus (to a maximum of four disks) or across SCSI buses.

Besides interconnecting nodal computers by an ATM network, the configuration can consist of processor cards interconnected via an I/O bus such as VME. In either case, the initial **open** request from the client first goes to the administrator node. This node then determines which of the server nodes has the requested media object and forwards the request to this server. From then on, communication is direct between a particular server node and the client.

Our architecture has the following features:

1. **Scalability:** The performance bottleneck of a CMFS is the I/O bandwidth. This is substantiated by the following speed differences: the disk system (typically between 2 to 5 MByte/sec), the I/O bus (SCSI-2 at 20 MByte/sec), the internal bus (800 to 1,200 Mbit/sec), and an ATM network (100 to 155 Mbit/sec). Given that the typical bandwidth required to support

a client ranges from 1 Mbit/sec to 8 Mbit/sec, the number of concurrent clients may be quite limited. However, our CMFS architecture is scalable, permitting the use of multiple server nodes as more capability is needed. Since the server nodes are independent of each other, the architecture does not impose any limit on the number of servers that can be added.

2. **Multimedia support:** Our CMFS design is not restricted to a single media syntax such as MPEG-2. A suitable abstraction for time and for the media units per second has been developed. This would effectively support the possibility of displaying the same video stream with a variety of different audio streams, where each stream may come from a different server.

3. **Disk I/O bandwidth:** In our system, the disk capability of the server is determined dynamically by calibrating the disk I/O bandwidth. This is more accurate than other studies which assume a static disk layout [5, 15]. Two values are determined in this calibration: the maximum and the minimum number of I/O operations per second. These values include the hardware overhead in transferring disk blocks as well as the operating system software overhead. Our design allows us to obtain a more realistic figure on the capacity of the server.

4. **Variable bit rate I/O scheduling:** A novel I/O scheduling algorithm based on variable bit rate (VBR) streams has been developed. This permits the scheduling of streams that have been compressed using VBR schemes such as motion JPEG and MPEG-2. An admission control algorithm and an I/O scheduler for variable bit rate traffic have been developed as part of the CMFS. A new stream is admitted if at any time instant, the combined data rate requirements of all streams do not exceed the I/O bandwidth. Our algorithm is more efficient when compared to algorithms based on constant bit rate traffic because a larger number of streams can be supported simultaneously.

4.2.2 Synchronization and QoS Support

The CMFS provides a programming interface that supports access to the media objects, synchronization, and QoS [9]. Referring to Section 3.5.2 where we presented the steps required for the presentation of a document. In steps 13 and 14, the synchronization module requests the CMFS to set up connections and deliver the media data. This is accomplished by using the **prepare** operation to request the CMFS to begin the data transfer of a media object. The **read** operation is then used to obtain data queued at the client. **read** is strictly a local client operation that does not result in a request to the CMFS. This "instantaneous" nature of **read**, coupled with the fact that there is a guaranteed bounded delay on **prepare**, supports the synchronization of multiple independent streams at the client even if the streams originate from different servers.

The CMFS is designed such that once **prepare** has returned control, the client is guaranteed to have sufficient data queued locally to support the continuous presentation of the media object. Underflow is therefore avoided. The **prepare** operation also has parameters which control the speed and amount of data that are transmitted. These parameters are used to vary the QoS.

4.3 Synchronization of Multimedia Data

We have designed and developed a novel algorithm to synchronize multiple media streams from possibly heterogeneous servers. Our algorithm adheres to the inter-media skew tolerances obtained by Steinmetz [26] which define the limits in the perception of ordinary human beings between various media types. For example, lip synchronization has a tolerance of 120 msec. In designing our algorithm, we adopt an approach that does not require a global clock among the various servers. Furthermore, the buffer requirements are kept to a minimum.

Our synchronization algorithm forms the core of the synchronization module. Referring again to the document presentation example in section 3.5.2, the synchronization module is involved in steps 11 to 15. More precisely, the synchronization module queries the database for meta-data such as length of document, frame rate, HyTime parameters, and the UOIs used to locate the media objects in the CMFS. These meta-data are then used to construct a presentation scenario, which segments the media streams into small pieces (for example, segments of one second duration) and defines the temporal relationship among the segments.

Our synchronization algorithm is executed at two levels. At the first level, the decoding delays are estimated, and the Time Flow Graph method [18] is used to determines the times at which the media servers should start transmitting their respective media streams. At the client, a media synchronization controller (MSC) is activated for each media stream. The MSCs are responsible for opening and controlling transport connections. They read the required segments according to the presentation scenario so as to play out the multimedia document in synchrony. Sometimes, scheduling and predicting the traffic are not sufficient to maintain a simultaneous multi-stream delivery since the network may introduce random delays and losses, resulting in jitters and gaps within the data stream. Compensating for such errors is done at the second level of synchronization, which works as follows. If during a segment time interval, the video MSC receives data out of synchronization with the audio stream (for example, with skew greater than 120 ms), it informs the audio MSC of the actual time-skew. During the next one-second interval, both MSCs shift their data presentations by the previously encountered skew, thus recovering synchronization. Details of the synchronization control system and performance evaluation results are reported in [17].

4.4 QoS Negotiation and Adaptation

The overall goal of QoS negotiation is to optimize the system configuration that can satisfy the users' QoS constraints. A framework for QoS negotiation has been defined which includes all system components such as the client workstation, network, and servers [27, 12]. The global configuration involved in a given instance of an application can be selected based on the user's QoS requirements and the resource availability at the different system components. For access to multimedia documents, the system may take advantage of the presence of several media variants [29]. For presentation to a specific user, the system selects the most appropriate media variant depending on the QoS preferences of the user (including cost) and the current availability of system resources. This selection involves the evaluation of various configuration alternatives. If the negotiated QoS cannot be maintained during the presentation of the document, possibly due to network or server congestion, the QoS manager may perform an automatic reconfiguration in order to maintain the originally agreed QoS characteristics.

Based on the above framework, a protocol for QoS negotiation has been designed and implemented. The details of this protocol were not included when we discussed the document presentation example in section 3.5.2. In that example, we have assumed that there is a match between the QoS variant found in the document meta-data and that contained in the user profile (see step 9). On the other hand, if there is no match, the protocol proceeds as follows.

A user may define different QoS profiles, each containing a set of selection criteria [12]. For each relevant QoS parameter, the criteria may include a minimum value and a preferred value. A priority ordering based on these parameters is also provided, either in absolute terms or in terms of a weighted sum. The latter is important because some kind of trade-off may be performed between conflicting preferences, such as low cost and high presentation quality. A match is not found if the system cannot provide a configuration which satisfies the minimum requirements of the user. In this case, the user is invited to accept certain quality reductions based on the feasible system configurations. The user may accept this alternative or simply abort the negotiation.

4.5 Scalable Video Encoding

Scalable video encoding [11] is an important feature in the design of distributed multimedia applications, because it provides efficient support for video objects at multiple levels of resolution. One can then accommodate requests to display video objects on terminals with different capabilities, and the transport of video over networks with a range of QoS availability.

Three types of scalability are commonly identified: spatial scalability, where the different levels have different spatial resolution; signal-to-noise ratio (SNR) scalability, where the different levels have the same spatial resolution but different amplitude resolution (or SNR); and temporal scalability, where the different levels have different temporal resolution. For the MPEG-2 standard, spatial scalability is the most relevant for meeting our objectives because it allows us to support receivers with different display capabilities. MPEG-2 provides for two levels of resolution in the spatial scalable extension, and we have developed such a two-level encoder/decoder in software. The original sequence at the higher level is first filtered and down-sampled to produce the low resolution picture.

This picture is then encoded using the appropriate MPEG-2 configuration, and the corresponding bit stream is stored or transmitted. At the decoder, the encoded low resolution picture is decoded and up-converted to the original resolution. The up-converted picture is then available to assist in encoding the original full resolution picture. The prediction of a picture in the high resolution sequence is formed using either the previous (and/or subsequent) high-resolution reference picture(s), the up-converted picture from the low level (at the same time instant) or a combination of the two.

4.6 Related Work

Enabling technology for distributed multimedia applications is an active area of research, and it is not practical in this paper to provide a survey of related work. Nevertheless, it is important to mention some related publications. These include [19, 22, 3] for multimedia data management, [14, 15] for continuous media file server, [4, 20] for QoS negotiation and adaptation, [10] for media stream synchronization, and [28, 13] for scalable video encoding. In addition, the recently announced ObjectStore version 5 has features that support multimedia, and the Internet Engineering Task Force has developed protocols such as RSVP [8] and RTP [7] that support the transport of media streams over the Internet.

5 Concluding Remarks and Future Direction

In this paper, we have described the software technologies developed by the CITR Broadband Services major project. Our technologies have a number of salient features, which are not present in most other systems:

- Our continuous media file server is scalable without the need for special hardware; it also supports QoS, variable bit rate transfer, and synchronization of media streams.
- Our database is based on an object-oriented design with an efficient storage structure, a uniform treatment of multiple media and meta-data, and a database model that is compliant with the SGML/HyTime standard.

- Our QoS management framework supports a dynamic choice of available services; that is, it selects an optimal configuration of the system components based on factors such as cost and resource availability.
- Our media synchronization algorithm is based on the time flow graph approach and does not require a global clock.

Our approach of organizing the constituent projects according to system components has worked very well. An important success factor is the close collaboration among members of the integration team. Our integration effort has led to improved understanding of the research issues related to each system component. Some of these issues might not have surfaced if the research had focused on a specific component only. We now have a testbed that can be used for research and development work in distributed multimedia applications.

As to future directions, we have recently started work on extending our technologies to include a conversational capability. This would allow users to engage in video-conferencing, and at the same time, access multimedia documents from a multimedia database. Such a capability would effectively support applications such as tele-learning and remote consultation. The conversational capability, together with a tele-learning application, will be developed.

6 Acknowledgements

We gratefully acknowledge the contributions made to this project by many people other than the authors, including K. Bennet, J. Konrad, M. Daami, R. Dssouli, S. El-Medani, D. Finkelstein, J. Gecsei, M. Ito, J. Jarmasz, L. Li, Y. Liu, C. Louvard, R. Mechler, R. Ng, S.-L. Ooi, S. Panchananathan, W. Robbins, M. Schöne, R. Somalingam, C. Vittel, and A. Vogel.

This work was supported by a grant from the Canadian Institute for Telecommunications Research under the NCE program of the Government of Canada, and by the IBM Toronto Laboratory Centre for Advanced Studies.

References

[1] ISO (20744). Hypermedia/time-based structuring language: Hytime. International Standards Organization, 1992.

[2] ISO (8879). Information processing - text and office information systems - standard generalized mark-up language. International Standards Organization, 1986.

[3] Thuraisingham B. On developing multimedia database management systems using the object-oriented approach. *Multimedia Review*, 3:11–19, 1992.

[4] A. Campbell, G. Coulson, and D. Hutchison. A quality of service architecture. *ACM Computer Communication Review*, 1994.

[5] E. Chang and A. Zakhor. Cost analysis for vbr file servers. *IEEE Multimedia*, 3(4):56–71, 1996.

[6] S. El-Medani. Support for document entry in the multimedia database. Master's thesis, Department of Computing Science, University of Alberta, 1996. Also available as Technical Report 96-23 (`http://ftp.cs.ualberta/pub/TechReports/TR96-23`).

[7] H. Schulzrinne et.al. Rtp: A transport protocol for real-time applications. Internet RFC 1889, Jan. 1996.

[8] R. Braden et.al. Resource reservation protocol (rsvp) - version 1 functional specification. Internet Draft draft-ietf-rsvp-spec-14, Nov. 1996.

[9] D. Finkelstein, N. C. Hutchinson, D. J. Makaroff, R. Mechler, and G. Neufeld. Real time threads interface. Available from: `ftp://ftp.cs.ubc.ca/pub/local/CITR/UBC/rtmanual.ps`.

[10] N. D. Georganas, R. Steinmetz, and T. Nakagawa, editors. *IEEE Journal on Selected Areas in Communications: Synchronization Issues in Multimedia Communications.* January 1996. Volume 14, Number 1.

[11] B. Girod. Scalable video for multimedia workstations. *Comput. & Graphics*, 17(3):269–276, 1993.

[12] A. Hafid and G. v. Bochmann. Quality of service adaptation in distributed multimedia applications. To be published in ACM Multimedia Systems Journal.

[13] T. Hanamura, W. Kameyama, and H. Tominaga. Hierarchical coding scheme of video signals with scalability and compatibility. *Signal Processing: Image Communication*, 5:159–184, 1993.

[14] R. Haskin and F. Stein. The tiger shark file system. In *IEEE 1996 Spring COMPCON*, February 1996.

[15] M. Kumar, J.L. Kouloheris, M.J. McHugh, and S. Kasera. A high performance video server for broadband network environment. In *SPIE'96, Multimedia*, 1996.

[16] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. The ObjectStore database system. *Communications of the ACM*, 34(10):50–63, October 1991.

[17] L. Lamont, L. Li, R. Brimont, and N. D. Georganas. Synchronization of multimedia data for a multimedia news-on-demand application. *IEEE Journal on Selected Areas in Communications: Synchronization Issues in Multimedia Communications*, 14(1):264–278, January 1996.

[18] L. Li, A. Karmouch, and N. D. Georganas. Multimedia teleorchestra with independent sources: Part 2 – Synchronization algorithms. *ACM/Springer Multimedia Systems Journal*, 1(4):154–165, February 1994.

[19] C. Meghini, F. Rabitti, and C. Thanos. Conceptual modeling of multimedia documents. *IEEE Computer*, 24:23–30, 1991.

[20] K. Nahrstedt. *An Architecture for End-to-End Quality of Service Provision and its Experimental Validation.* PhD thesis, University of Pennsylvania, 1996.

[21] S.-L. Ooi. An api for distributed multimedia applications. Master's thesis, Department of Computer Science, University of Waterloo, 1995.

[22] E. Oomoto and K. Tanaka. Ovid: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Management*, 5:629–643, 1993.

[23] M. T. Özsu, S. El-Medani, P. Iglinski, M. Schnöe, and D. Szafron. An object-oriented SGML/HyTime multimedia database management system. Submitted for publication, 1996.

[24] M.T. Özsu, D. Szafron, G. El-Medani, and C. Vittal. An object-oriented multimedia database system for a news-on-demand application. *Multimedia Systems*, 3:182–203, 1995.

[25] M. Schöne. A generic type system for an object-oriented multimedia database system. Master's thesis, Department of Computing Science, University of Alberta, 1996. Also available as Technical Report 96-14 (`http://ftp.cs.ualberta/pub/TechReports/TR96-14`).

[26] R. Steinmetz. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communications: Synchronization Issues in Multimedia Communications*, 14(1):61–72, January 1996.

[27] G. v. Bochmann and A. Hafid. Some principles for quality of service management. In *Proceedings of the Fourth International Workshop on Quality of Service (invited paper)*. Paris, France, March 1996. Revised version to be published in the Distributed Systems Engineering Journal.

[28] M. Vetterli and K. Uz. Multiresolution coding techniques for digital television: a review. *multidimensional Systems and Signal Processing*, 3:161–187, 1992.

[29] A. Vogel, B. Kerhervé, G. v. Bochmann, and J. Gecsei. Distributed multimedia applications and quality of service: A survey. *IEEE Multimedia*, 2(2):10–19, 1995. A reduced version was published in the CASCON'94 Proceedings, October 1994.

Appendix A: Sample Document Markup

```
<!doctype article SYSTEM ”newsMM.dtd”>
<article id=”CITRNews-RV-01” language=”English”>
<frontmatter>
<edinfo>
<author>David Evans</author>
<date>11/15/1996</date>
<keywords>CITR, IBM, CAS, CASCON, Events, Conference.</keywords>
<loc>Toronto</loc>
<subject>Report on CITR BroadbandServices participation at CASCON ’96</subject>
<source>CITR</source>
</edinfo>
<hdline>CITR Broadband Services Major Project Demo at CASCON ’96 a Success
    </hdline>
<subhdline>Attendees impressed with demonstration of results of CITR
    Broadband Services Project.</subhdline>
<abs-p>
<paragraph>
The CITR Broadband Services Major Project presented its latest
work at the CASCON ’96 conference in Toronto.
</paragraph>
</abs-p>
</frontmatter>
<async>
<section>
<paragraph>
<figure>
    <image id=”CITR-news-image-1” variantspec=”image1-variant”>
</figure>
<figure>
    <image id=”CITR-news-image-2” variantspec=”image2-variant”>
</figure>
</paragraph>
</section>
<image-variant id=”image1-variant” filename=”images/citr_logo.gif”
    format=”gif” size=”525” width=”95” height=”64” color=”colour”>
<image-variant id=”image2-variant” filename=”images/cascon.gif”
    format=”gif” size=”476” width=”88” height=”80” color=”colour”>
</async>
<sync>
<audio-visual id=”audio-visual-1”>
    <x id=”x-1”>
    <y id=”y-1”>
    <time id=”time-1”>
    <av-fcs id=”fcs-1”>
        <av-evsched id=”evsched-1”>
            <audio id=”audio-1” price=”5” variantspec=”audio-variant-1 audio-variant-2” exspec=”extlist-1”>
            <video id=”video-1” price=”5” variantspec=”video-variant-1 video-variant-2” exspec=”extlist-1”>
        </av-evsched>
    </av-fcs>
    <av-extlist id=”extlist-1”>
```

```
        <xdimspec id=”xdimspec-1”>1 320</xdimspec>
            <ydimspec id=”ydimspec-1”>1 240</ydimspec>
            <tdimspec id=”tdimspec-1”>1 156</tdimspec>
        </av-extlist>
    <video-variant id=”video-variant-1”
        format=”mjpeg” streamspec=”video-stream-1” site=”bristol-atm”
        duration=”155” width=”320” height=”240” framerate=”5”
        bitrate=”20185” color=”color”>
    <video-variant id=”video-variant-2”
        format=”mjpeg” streamspec=”video-stream-2” site=”bristol-atm”
        duration=”155” width=”320” height=”240” framerate=”15”
        bitrate=”20185” color=”color”>
    <audio-variant id=”audio-variant-1”
        format=”g728” streamspec=”audio-stream-1” site=”bristol-atm”
        duration=”155” samplerate=”22050” bps=”16” quality=”CD”
        language=”English”>
    <audio-variant id=”audio-variant-2”
        format=”g728” streamspec=”audio-stream-2” site=”bristol-atm”
        duration=”155” samplerate=”22050” bps=”16” quality=”CD”
        language=”English”>
    <stream id=”video-stream-1” uoi=”1” size=”22050”>
    <stream id=”video-stream-2” uoi=”2” size=”100000”>
    <stream id=”audio-stream-1” uoi=”3” size=”22050”>
    <stream id=”audio-stream-2” uoi=”4” size=”322050”>
</audio-visual>
</sync>
</article>
```

About the Authors

Gregor v. Bochmann has been a professor at the Université de Montréal since 1972 and holds the Hewlett-Packard-NSERC-CITI chair of industrial research on communication protocols. He is also one of the scientific directors of the Centre de Recherche Informatique de Montréal (CRIM). He is a Fellow of the IEEE and ACM. Bochmann has worked in the areas of programming languages, compiler design, communication protocols, and software engineering and has published many papers in these areas. He has also been actively involved in the standardization of formal description techniques for OSI. His present work is aimed at methodologies for the design, implementation and testing of communication protocols and distributed systems. Ongoing projects include distributed systems management and quality of service negotiation for distributed multimedia applications.

Jeff Brinskelle received his honours B.C.S. degree from Carleton University in 1993. His Honours thesis was on Interactive Computer Music. He started working in the Multimedia Communications Research Laboratory (MCRLab) at the University of Ottawa in December 1994, as a software research engineer. His main focus is on the Multimedia Synchronization project which is part of the CITR Broadband Services major project. He is also MCRLab manager.

Eric Dubois received the B.Eng. (honours) degree with great distinction and the M.Eng. degree from McGill University in 1972 and 1974 respectively, and the Ph.D. from the University of Toronto in 1978, all in electrical engineering. He joined the Institut National de la Recherche Scientifique (University of Quebec) in 1977, where he currently holds the position of professor in the INRS-Telecommunications centre in Montreal, Canada. In 1994-95 he was with the Broadcast Technologies Research Directorate of the Communications Research Center in Ottawa, Canada. He is an associate editor of the IEEE Transactions on Image Processing and a member of the editorial board of the EURASIP journal Signal Processing: Image Communication. He was co-guest editor of the June 1994 issue of that journal, a special issue on motion estimation and compensation technologies for standards conversion. His research has centered on the source coding and processing of still and moving images, and in multidimensional digital signal processing. He is a Fellow of the

IEEE and a member of the Order of Engineers of Quebec.

David Evans is a research associate at the University of Waterloo. He received his Masters of Mathematics in Computer Science from the University of Waterloo and his Bachelor of Science in Computing and Information Science from the University of Guelph. His research interests include self-organization, performance analysis, and security as applied to broadband networks and applications.

Nicolas D. Georganas is Professor of Electrical and Computer Engineering, University of Ottawa. He has been a faculty member in that Department since 1970 and served as Chairman from 1981 to 1984. From 1986 to 1993, he was Dean of the Faculty of Engineering. He has published over 200 technical papers and is co-author of the book "Queueing Networks - Exact Computational Algorithms: A Unified Theory by Decomposition and Aggregation", MIT Press, 1989. He is General Chair of the IEEE Multimedia Systems '97 conference, and was Co-Chair of the Canadian Conference of Electrical and Computer Engineering in 1990. He has served as Guest Editor of the IEEE Journal on Selected Areas in Communications, issues on "Multimedia Communications" (April 1990) and on "Synchronization Issues in Multimedia Communications" (1995) and as Technical Program Chair of MULTIMEDIA '89, the 2nd IEEE COMSOC International Multimedia Communications Workshop in 1989, and the ICCC Multimedia Communications '93 Conference. He is on the Editorial Boards of Performance Evaluation, Computer Networks and ISDN Systems, Computer Communications, Multimedia Tools and Applications, and was an editor of the IEEE Multimedia Magazine. He is an IEEE Fellow, a Governor of the ICCC, and a Fellow of the Engineering Institute of Canada. In 1995, he was co-recipient of the IEEE INFOCOM '95 Prize Paper Award. His current research interests are in broadband multimedia communications and internet tele-collaboration tools.

Abdelhakim Hafid is a Researcher Staff Member at the Computer Science Institute of Montreal, Telecommunications and Distributed Systems Division, working in the area of distributed multimedia applications. He received his M.Sc. with first class honours in computer engineering from EMI, Rabat, Morocco in 1991. He received his Ph.D. degree in computer science from the Université de Montréal on quality of service negotiation and adaptation in 1996. During the year

1993-1994 he worked as a guest scientist at GMD-FOKUS, Systems Engineering and Methods group, Berlin, Germany in the area of high speed protocols testing. His current research interests are in broadband multimedia services and communications.

Norman Hutchinson received the B.Sc. degree from the University of Calgary in 1982, and the M.Sc. and Ph.D. degrees in Computer Science from the University of Washington in 1985 and 1987 respectively. After graduation he was an assistant professor of Computer Science at the University of Arizona, Tucson, Arizona for 4 years before moving to the University of British Columbia in 1991. His research interests are centered around programming languages and operating systems, with particular interests in object-oriented systems and distributed systems. He was instrumental in the design of the Emerald distributed programming language and the x-kernel communication protocol environment. This work has spawned a new operating system research project, called Kea, which is exploring techniques for embedding application specific policies and mechanisms in operating systems. Recently he has been additionally working in the area of soft real time systems and is currently developing a multimedia file server as a platform for experimenting with these ideas.

Paul Iglinski is a Research Associate with the Multimedia Data Management Project in the Department of Computing Science of the University of Alberta. He has an M.Sc. (1994) in Computing Science from the University of Alberta.

Brigitte Kerhervé is an associate professor in the Computer Science Department at Université du Québec e Montréal. She received her Ph.D. degree in computer science from the University of Paris VI, France in 1986. From 1982 to 1987, she participated in the Sabre project at Institut National de Recherche et Automatique (INRIA, France), on the design and implementation of a complete database system. From 1987 to 1989, she was involved in European research projects in the field of advanced database systems. From 1989 to 1992, she has been an assistant professor at Ecole Nationale Superieure des Telecommunications, Paris, France. Her research interests include metadata for distributed multimedia systems, object and distributed database systems and quality of service management.

Louise Lamont received the B.A.Sc. degree in Electrical Engineering from the University of

Ottawa, Canada, in 1977. Through her extensive work experience at companies such as Gandalf Data and Bell-Northern Research, she acquired comprehensive design expertise in data communication protocols, telecommunications, software environments and multimedia applications. She was a research associate in the Multimedia Communications Research Laboratory (MCRLab), University of Ottawa, involved in the study and design of various multimedia applications. She is now with the Communications Research Centre of Industry Canada.

Kelly Lyons is a Research Staff Member at the Centre for Advanced Studies in the IBM Software Solutions Toronto Laboratory working in the area of distributed multimedia applications. She received her B.Sc. in Computing and Information Science from Queen's University in 1985 at which time she started working in Compiler Development at IBM Canada Ltd. Laboratory. In 1987, she was granted Educational Leave of Absence from IBM and returned to Queen's. In 1988, she received her M.Sc. in Computing and Information Science from Queen's University in the area of computational geometry. She received her Ph.D. degree from the Department of Computing and Information Science at Queen's University on graph layout algorithms in 1994. In 1996, she was appointed to the position of Adjunct Professor at York University. Her research interests include distributed multimedia applications, ATM networks, distributed computing, database management systems, digital libraries, graph layout algorithms, and computational geometry. She can be reached at the IBM Toronto Laboratory, 2G/894, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7, email: klyons@vnet.ibm.com.

Dwight Makaroff is a Ph.D. student in Computer Science at the University of British Columbia. His research interests include distributed multimedia systems, video servers, and operating systems. He is a major contributor to the design and implementation of the UBC continuous media file server. Makaroff received his B.Comm. and M.Sc. degrees in Computational Science from the University of Saskatchewan in 1985 and 1988, respectively. He has previously been a faculty member at Bethel College and Trinity Western University.

Gerald Neufeld received the B.Sc. degree and M.Sc. from the University of Manitoba in 1976. In 1979 he received a M.Dip in Christian Studies from Regent College at the University of British

Columbia. Neufeld received the Ph.D. degree in Computer Science from the University of Waterloo in 1986. After graduation he became an assistant professor of Computer Science at the University of British Columbia. He is now an associate professor at the same institution. Neufeld's research interests include networking, distributed systems and operating system support for networking and distributed systems. He has worked on the CMFS project for the past two years. Other related work includes the development of a shared memory multiprocessor kernel for high-speed networking, a Java-based multimedia document system, distributed directory services and application-level multicast.

M. Tamer Özsu is a Professor of Computing Science at the University of Alberta. He received his Ph.D. (1983) in Computer and Information Science from the Ohio State University. His research interests are on distributed database, distributed object management, multimedia information systems and interoperability issues. He has authored or co-authored four books and a number of technical papers in these areas. He is on the editorial boards of The VLDB Journal, Distributed and Parallel Databases, and Parallel & Distributed Technology. He also serves on the Board of the VLDB Endowment.

Duane Szafron is an Associate Professor of Computing Science at the University of Alberta where he is working on research in the areas of object-oriented computing, parallel and distributed systems, graphical user interfaces and multimedia. He holds a B.Sc. in Physics and Mathematics and an M.Sc. in Mathematics from the University of Regina and a Ph.D. in Applied Mathematics from the University of Waterloo. As a consultant for the British Columbia Ministry of Crown Land, he was the leader of the three-person team that created the initial design of the SAIF object-oriented specification language for geographic information systems. This language has now become a Canadian standard for geomatics. Dr. Szafron is also the designer of the CHILDS library system that is used by more than 200 school libraries.

Rolf Velthuys worked on his Ph.D. research at the IBM European Networking Center (ENC) in Heidelberg, Germany and completed a degree from Berne University in 1992. His research was on generation of test descriptions based on formal system specifications. He taught at the University of

British Columbia as a Post Doctoral Fellow until 1994 when joined the CITR Broadband Services project where he held a Post Doctoral Fellowship jointly from the University of Waterloo and IBM Canada, Ltd. Velthuys spent most of his time at the IBM Centre for Advanced Studies in Toronto where he was responsible for integrating the work of the various subprojects. In July 1996, he joined KPN Research in the Netherlands as a technical consultant. His research interests are communication aspects of distributed applications, and the design and development of new distributed applications which exploit available new technology.

Johnny W. Wong received the B.S. degree in engineering, and the M.S. and Ph.D. degrees in computer science from the University of California at Los Angeles in 1970, 1971, and 1975, respectively. Since 1975, he has been with the University of Waterloo where he is currently a professor of computer science. In September 1994, he completed a five-year term as Associate Provost, Computing and Information Systems. He was a visiting scientist at the IBM Zurich Research Laboratory from September 1981 to August 1982, from September 1988 to August 1989, and from September 1995 to August 1996. He was Editor for Wide Area Networks for the *IEEE Transactions on Communications* from 1989 to 1992, and served on the Editorial Board of *Performance Evaluation* from 1986 to 1993. He was Technical Program Chair of IEEE INFOCOM '84 and of the 1994 International Conference on Computer Communications and Networks. His research interests include network resource management, distributed multimedia applications, and performance evaluation.