Enabling Technology for Distributed Multimedia Applications

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

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

J. Brinskelle, D.F. Evans, A. Hafid, N. Hutchinson, P. Iglinski,

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

February 16, 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 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 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. In this paper, the reference architecture, research results, and prototyping effort are presented.

This paper is organized as follows. In Section 2, we discuss the key design decisions and present an overview of our reference architecture. The organization of the Broadband Services research program is also described. In Sections 3 to 8, we consider each constituent project in turn, and present our approach and accomplishments. Finally, Section 9 contains some concluding remarks and a discussion of future research directions.

2 Reference Architecture 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 design decisions:

1. **Uniform Treatment of Content Data and Meta-Data:** In a multimedia document, the *content data* corresponds to the text, image, audio, and video data, and *meta-data* contains descriptive information about the content data. Meta-data includes 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 monomedia objects. 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 DBMS 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 comes 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 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 monomedia 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 can 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 monomedia 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 conditions as well as varying presentation quality requested by the users. The latter is relevant,

[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.

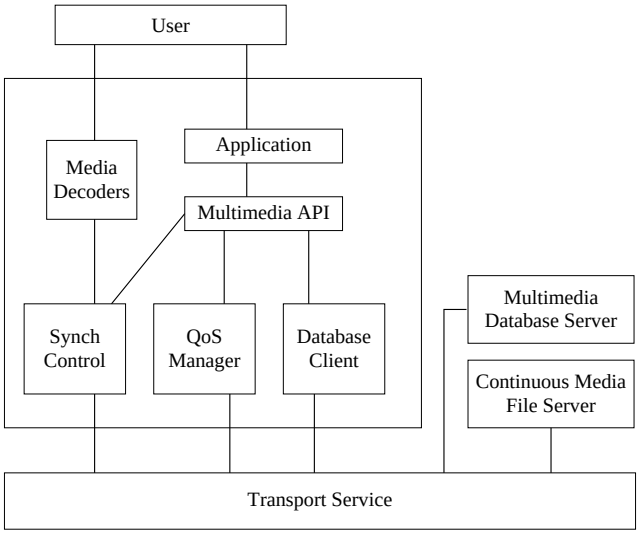

Figure 1: Reference Architecture

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 Reference Architecture

Our reference architecture, developed as a result of our design decisions, is shown in Figure 1. From the viewpoint of the user, the system is a distributed multimedia DBMS which provides facilities to browse, search, and retrieve multimedia documents. The DBMS is based on a client-server model, and we will refer to the client and server components of the database as *client DBMS* and *server DBMS* respectively. Queries issued by a user are processed by the client DBMS. The client DBMS communicates with the server DBMS to retrieve the meta-data regarding the requested multimedia documents. The meta-data contains, among other parameters, the server locations of the various

monomedia objects of the document, the QoS information (for example, encoding scheme or frame rate for video), and the *presentation scenario* which describes the temporal relationships of these objects. The client DBMS passes the retrieved meta-data to the QoS manager, which determines the most suitable QoS variant to be presented to the user. This information is passed to the scheduler (a component of the synchronization control module), which builds a *presentation schedule* for each selected monomedia object. Synchronization control then requests the database and continuous media file servers to transmit the required data across the network, and coordinates the reception of data streams at the client for presentation to the user.

From the viewpoint of a service provider, facilities are required to create multimedia documents and enter these documents into the database. Facilities are also required for document update. We distinguish between version update and content update. Version update is appropriate for applications such as multimedia news where an existing news item is replaced by a more up-to-date version of the same story. Content update means changing the content of an existing document. Our work is concerned with version update only because multimedia news has been selected as our target application.[2] To perform an update, the client DBMS communicates with the server DBMS to enter the updated version into the database. The existing version is marked out-dated, and will no longer be available to the user.

Finally, an application programming interface (API) is defined to facilitate application development.

2.3 Project Organization

The system components can readily be identified from the reference architecture shown in Figure 1. These components are being investigated in six constituent projects:

[2] Content update will be considered as a future research activity.

1. **Multimedia Data Management:** This project is led by M. T. Özsu of the University of Alberta. It is concerned with the logical modeling of multimedia data, the design of query languages, content-based indexing of images, and the storage, retrieval, and update of multimedia documents.

2. **Continuous Media File Server:** This project is led by G. Neufeld of the University of British Columbia. It is concerned with the design and development of a scalable continuous media file server as well as the provision of real time support by the server and client operating systems, and by the transport network.

3. **QoS Negotiation and Adaptation:** This project is led by G. v. Bochmann of the Université de Montréal. It is concerned with QoS negotiation between the user and the system, the adaptation of an application to changing QoS, and the monitoring of system and network performance for QoS negotiation purposes.

4. **Synchronization of Multimedia Data:** This project is led by N. D. Georganas of the University of Ottawa. It is concerned with the design, implementation, and performance evaluation of scheduling algorithms to synchronize multiple media data streams at the client according to a given presentation scenario.

5. **Scalable Video Encoding:** This project is led by E. Dubois of INRS Telecommunications. It is concerned with scalable video coding/decoding schemes. Such schemes can be used to effectively support different levels of resolution in video objects.

6. **Project Integration:** This project is led by J. Wong of the University of Waterloo. It is concerned with system integration where a prototype is constructed using the software modules developed by the various projects. Included in this activity is the development of the API and a multimedia news application.

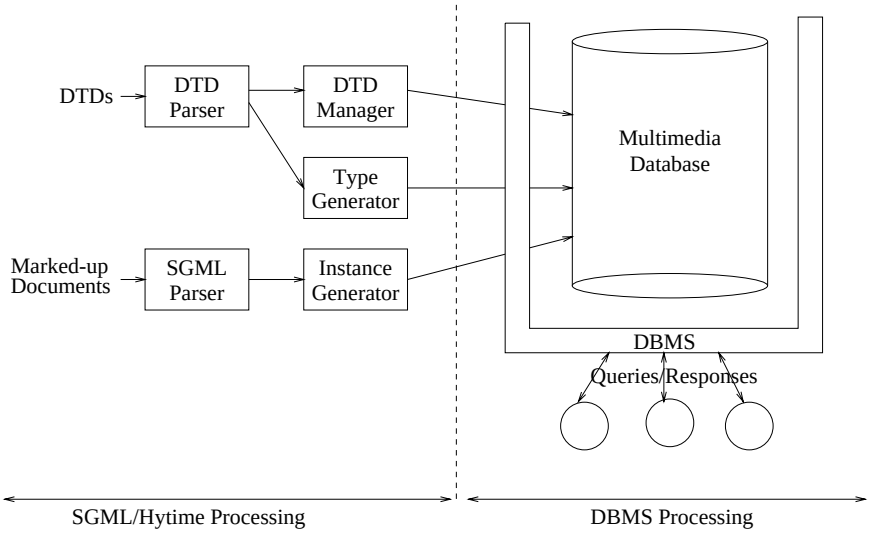

Figure 2: Multimedia DBMS

Finally, 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 activity is to coordinate the milestones of the constituent projects so that the objectives of the major project are met.

3 Multimedia Data Management

The multimedia DBMS is an object-oriented system that complies with the SGML/HyTime standard [2, 1] (see Figure 2). The system is capable of storing different types of documents within one database by accommodating multiple document type definitions (DTDs). Objects are dynamically created according to element definitions in a given DTD. The system also has tools to automatically insert marked-up documents into the database.

3.1 Multimedia Support

The system is built as an extension layer on top of a generic[3] object DBMS called ObjectStore [10]. The extensions include specific support for multimedia information systems. The most important part of the extension layer is the type system. In this context, there are three fundamental issues. First, the basic monomedia components of the document (that is, text, image, audio, and video) need to be modeled. Secondly, to support the multimedia news target application, the structure of the news documents must be represented. Third, meta-data about the multimedia objects and applications have to be captured and stored in the database. Since ObjectStore does not provide native support for basic multimedia data other than text (i.e., strings), the type system defines these data types and refers to them as *atomic types*. Currently, these atomic types store meta-data about all media types as well as image and text objects. Audio and video objects are stored in a continuous media file server. The multimedia DBMS provides a uniform interface to these repositories.

3.2 Document Representation

We follow the SGML/HyTime standard for representing document structure. SGML formally specifies this structure by defining element types such as paragraph and figure and the relationships between them in a 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. For example, a single `Book` DTD, hierarchically composed of chapter, section, paragraph, and word elements, might serve as the template for many instances of book.

The representation of spatio-temporal relationships between monomedia objects is an important consideration in designing a multimedia database. Such information is required by the scheduler to plan the retrieval and presentation of multimedia documents. In following the HyTime philosophy,

[3] In the sense that it does not have native multimedia support.

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. This is accomplished using individualized style sheets which are stored in the database as objects. The second and arguably more important consideration is to represent the spatio-temporal relationships in accordance with the HyTime standard.

HyTime defines a number of architectural forms to deal with various hypermedia concepts. One of these architectural forms is the finite coordinate space (FCS) which is used to model spatio-temporal relationships. An FCS is a set of axes in finite dimensions. The units of measurement along these axes are called quanta. There are various types of quanta defined in HyTime along with the normal units of measurement such as characters, words, and nodes in trees. 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 the various axes defining the FCS form an *extent* which corresponds to an *event* [1]. An event schedule, consisting of one or more events, is used to represent temporal relationships among various monomedia objects. Within this context, our model of spatio-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 [19, 18].

3.3 Dynamic Insertion of New DTDs

A fundamental requirement of a multimedia DBMS for SGML/HyTime documents is that it should be able to handle multiple DTDs and support the creation of types that are induced by these DTDs. This is essential if the multimedia DBMS is to support a variety of applications. The system must analyze new DTDs and automatically generate 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."

In our system, 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 in the DTD. This object contains information about the element, such as its name, attribute 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. There are two important problems that need to be addressed in this process. Both of them are abstraction problems that can 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 should be 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. This factoring must be done automatically. If the feature is a common component, this is straightforward; otherwise, the problem is harder to solve.

Second, common element definitions across different DTDs should be represented by a common type in the type system. There is no easy solution to this problem since it leads to the well-known semantic heterogeneity problem that has been studied extensively within the multi-database community. Briefly, the problem is one of being able to determine whether two elements are semantically equivalent. Since this is not a trivial problem, we have chosen to give up some abstraction in favor of a semantically "safe" type system. Further details of handling multiple DTDs can be found in [20].

3.4 Type Re-use across DTDs

The complexity of the semantic heterogeneity problem does not mean that we completely abandon type re-use across DTDs. Our approach is to re-use atomic types such as `Image`, `Text` and high-level abstract supertypes such as `TextElement`, `Structured`, and `HyElement`. These types are safe to re-use because they have well-defined semantics and appear across many document types. 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.

3.5 Automatic Insertion of Multimedia Documents

A short-coming of many multimedia DBMS's is the lack of tools for the insertion of documents into the database. This is generally considered to be outside the scope of database work. In our investigation, we couple the multimedia database with a retrofitted SGML parser[4] [5]. SGML documents can then be created using existing authoring tools and automatically inserted into the database. The SGML parser accepts an SGML document instance from the authoring tool, validates it, and forms a parse tree. Another software module, called the instance generator, traverses the parse tree and instantiates the appropriate objects in the database. These are persistent objects, and can be accessed using the query facility.

4 Continuous Media File Server

Our continuous media file server (CMFS) is a specialized file server designed to support continuous media objects such as video and audio [3, 16]. The motivation for designing a specialized file server for continuous media is now well established. Data access patterns, as well as the services provided to the clients by such a server, differ considerably from those of a conventional distributed service such as NFS. A continuous media client typically transfers large volumes of sequential data, and the required resources at the network and at the server differ considerably. To ensure continuous data

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

transfer, the allocation of resources such as disk and network bandwidth, as well as processor cycles and memory at the server, must be guaranteed.

Many file servers assume that each individual stream has a constant bit rate (CBR). Compression methods such as motion JPEG or MPEG-2 produce streams whose bit rates vary considerably. It is, of course, possible to produce a constant bit rate stream with these schemes, but doing this would either increase the resource requirements or impair the quality of the stream. Data networks are well-suited to carry bursty traffic. In particular, much effort has gone into making ATM networks capable of handling bursty data sources. It is therefore reasonable to design a file service which can explicitly accommodate such variation in resource requirements and thereby increase the number of simultaneous streams that can be supported.

4.1 Design Features

The CMFS addresses the issues mentioned above through four major design 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 by a continuous media client ranges from 1 Mbit/sec to 8 Mbit/sec, the number of concurrent clients may be quite limited. The service must therefore be scalable, permitting more components to be added as more capability is needed. Section 4.2 describes the scalable architecture of our system.

2. **Multimedia support:** A client typically requires multiple concurrent media streams. For example, a session may include video, audio and captioned text. In order to support our design decision that the same video may be displayed together with a variety of different audio streams and that each stream may come from a different server, the server should not be restricted to a single media syntax such as MPEG-2. It must provide a suitable abstraction for time and

for media units per second.

3. **Disk I/O bandwidth:** Much effort has been expended in determining the optimal disk performance and its relationship to the number of clients which may be supported. Most analyses, however, assume a static disk layout [4, 13]. In our system the disk capability of the server is determined dynamically by calibrating the disk I/O bandwidth. This calibration determines two values: the minimum and the maximum 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. **VBR scheduling:** The I/O scheduling is based on variable bit rate (VBR) streams rather than constant bit rates. This permits the scheduling of streams that have been compressed using VBR schemes such as motion JPEG and MPEG-2. The variation does not have to be localized within a set of I, B and P frames; it can last for seconds across video scene boundaries. The CMFS incorporates the design of both an admission control algorithm and a stream scheduler for variable bit rate traffic [14, 15, 16].

4.2 CMFS Architecture

The design of 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. The server nodes are connected to an ATM network for delivering continuous media data to the client systems (see Figure 3). 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.

Multiple server nodes can be configured to increase the capability of the service. Since the server nodes are independent of each other, any number can be added. Besides interconnecting nodal

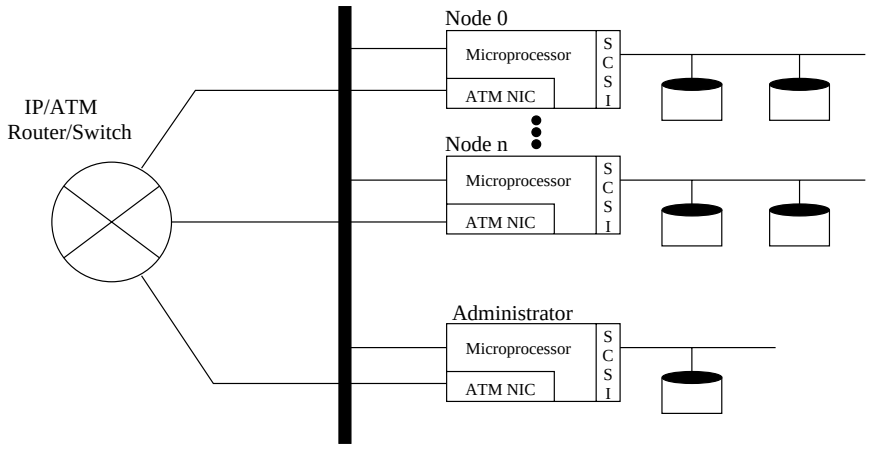

Figure 3: Continuous Media File Server Architecture

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 stream and forwards the request with additional information about the stream's playout. From then on, communication is direct between a particular server node and the client.

4.3 QoS and Synchronization Support

The CMFS provides a programming interface to the other system modules [6]. This interface is designed to support access to the monomedia objects as well as QoS and synchronization requirements. Using a *push* model of transmission, the **prepare** operation requests the server to begin data transfer of a monomedia object. The client then uses the **read** operation 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 interface 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 monomedia object. Underflow is therefore avoided. The **prepare** operation also has parameters which control the speed and amount of data that is transmitted. These parameters are used to vary the quality of service.

5 QoS Negotiation and Adaptation

We have developed a novel model for QoS negotiation in distributed multimedia systems. This model is based on the premise that advanced distributed applications must be adaptive in the sense that they must cope with variations in network conditions and with varying demands from users on the quality of the requested multimedia materials. The overall goal of QoS negotiation is to optimize the system configuration that can satisfy the users' QoS constraints. There are three major parties involved in our negotiation model: the user (at the client), the transport system, and the database that stores the multimedia documents. The latter is included because our distributed database system provides support for different variants of audio or video objects - each may be at a different resolution, and stored in a different server. The negotiation is performed by a QoS management module [22].

5.1 QoS Framework

A framework for QoS negotiation has been defined which includes all system components such as the client workstation, networks, and servers [22, 9]. 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 [24]. Examples of media variants include video with a different number of frames per second, and images with a different number of pixels per row or column, different color quality, or different encoding schemes. For presentation to a specific user, the system selects the most appropriate media variant depending on the QoS preferences of

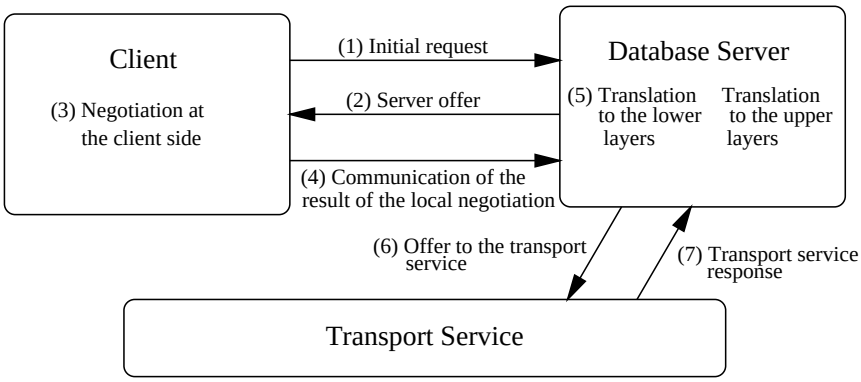

Figure 4: QoS Negotiation

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.

5.2 QoS Negotiation

The steps involved in QoS negotiation are shown in Figure 4. A key element in the negotiation process is the selection criteria provided by the user. Specifically, the user can define different QoS profiles, each containing a set of selection criteria [9]. 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 must normally be performed between conflicting preferences, such as low cost and high presentation quality. If the system cannot provide a configuration which satisfies the minimum requirements of the user, the user is invited to accept certain quality reductions based on the feasible system configurations.

QoS negotiation must cope with system components that provide QoS guarantees as well as

components that provide only best-effort service. In order to provide an end-to-end guarantee for QoS characteristics in the presence of best-effort components such as the Internet, the QoS management framework supports automatic QoS monitoring of those components that are known to be relatively unreliable. Our system includes a network performance monitor which may initiate a system reconfiguration if the measured performance falls below the QoS threshold which was determined during the initial configuration of the application.

An important aspect of this work is the identification of the system management information and the meta-data of the multimedia documents which are necessary for making sensible decisions concerning QoS negotiation. The management information is usually distributed among the different system components, while the document meta-data is stored in the database.

6 Synchronization of Multimedia Data

In general, multimedia synchronization denotes a temporal, spatial, or even logical relationship between objects or media streams. In the context of multimedia computing and communications, however, multimedia synchronization is typically concerned with temporal relationships only. This notion is still very broad and it captures a variety of issues such as inter-process communication. Recent advances in this field can be found in [7].

For our system, we have developed an algorithm to synchronize the various media streams from possibly heterogeneous servers. Our algorithm adheres to the inter-media skew tolerances obtained by Steinmetz [21] 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.

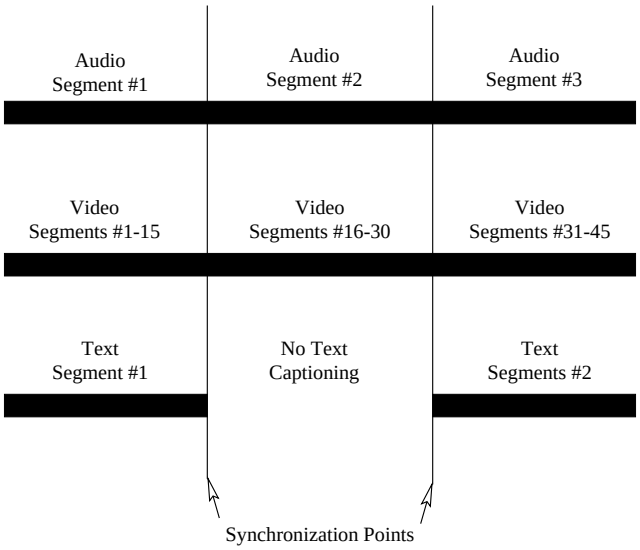

Figure 5: Presentation Scenario

6.1 Presentation Scenario

In our synchronization algorithm, control of the display of the media streams is done at the client. This is accomplished by the use of 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. An example is shown in Figure 5 where inter-stream synchronization is defined at synchronization points.

When a document is created, its presentation scenario is determined and stored as meta-data. At document presentation time, if a particular segment arrives too early, it may be intentionally delayed in a buffer in order to re-synchronize with the other media streams. If a segment arrives later than the acceptable skew tolerances, it is simply discarded. On the other hand, if the segment is late but falls within these tolerances, it is played as soon as possible.

6.2 Synchronization Control

Our synchronization algorithm is executed at two levels. At the first level, a scheduler (residing at the client) receives the presentation scenario from the database and the negotiated QoS parameters from the QoS manager. The scheduler then estimates decoding delays, and uses the Time Flow Graph method [12] to derive a presentation schedule for each monomedia object, that is, it determines the times at which the 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 request the start and end of data transfers according to the presentation schedule so that the multimedia document is played back in synchrony. 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.

The second level, which is called stream synchronization protocol, provides synchronization recovery operations at the MSCs. This protocol works as follows. The media streams are divided into segments of a given duration, say one second (see Figure 5). If during a one-second interval, the video MSC receives data out of synchronization with the audio stream (for example, with skew greater than 120 ms), it informs the other MSCs of the actual time-skew. During the next one-second interval, all 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 [11].

7 Scalable Video Encoding

In many applications, it may be expected that video objects can be displayed on a variety of terminals with different capabilities, ranging from portable personal computers to high-definition television

receivers. Furthermore, users may want to access the data at different bit rates, especially when the cost of receiving the data is sensitive to the bit rate used. The system should therefore provide the user with the ability to access a video sequence at different spatial and color resolutions, or at different bit rates.

The most straightforward approach to this is separate storage of the coded data for each type of receiver. This is complex and wasteful of storage capacity, and introduces the overhead of keeping several copies of the same object consistent. A more efficient approach is the concept of embedded coding, whereby a subset of the encoded data can be used to decode a lower resolution version of the sequence. This type of encoding method is referred to as *scalable encoding* [8]. 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.

The MPEG-2 standard has provided for spatial, SNR and temporal scalability in a limited fashion. Spatial scalability is the most relevant for the current application, with receivers having different display capability. 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.

8 Project Integration

The various system components have been implemented and successfully integrated into a multimedia news prototype. The integration effort is the responsibility of an integration team, led by R.J. Velthuys (September 1994 to June 1996) and D.F. Evans (since July 1996), and comprised of research staff and graduate students from all participating institutions. Integration was required 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 components. At the implementation level, integration team members interact frequently by email, phone call, and short term visits. System integration is done at the University of Waterloo. 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 multimedia news prototype.

As part of the project integration effort, a multimedia API has been defined and implemented, and a multimedia news application developed using this API. In the remainder of this section, we describe our API and then present the features of the multimedia news prototype.

8.1 Multimedia API

Table 1 contains an overview of the API primitives, organized by function groups [17, 23]. This is a minimal API, which has been implemented as a C++ class library [17], using the capabilities of the various system components.

8.2 Multimedia News Prototype

The latest version of our multimedia news prototype was demonstrated at the CASCON '96 conference. This version has the following features. The ObjectStore server and the CMFS reside on an

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 present pause fast forward/rewind close	prepare for retrieval; this includes the pre-fetching of data start or resume presentation 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

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.

The software technologies included in our integrated prototype are listed below:

- Document type definition, multimedia database, dynamic insertion of new types, reuse of types, and automatic insertion of documents.
- Real time threads, transport service based on UDP, and continuous media file server.
- QoS negotiation protocol, user QoS profile management, and QoS measurement layer.
- Synchronization control module.
- Multimedia API and multimedia news application.

9 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 multimedia 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 architecture 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.

10 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] S. Bikash, M. Ito, and G. Neufeld. Design and performance of a multimedia server for a high speed network. In *IEEE International Multimedia Conference*, May 1995.

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

[5] 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`).

[6] 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`.

[7] 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.

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

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

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

[11] 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.

[12] 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.

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

[14] D. Makaroff, G. Neufeld, and N. Hutchinson. An evaluation of VBR disk admission algorithms for continuous media file servers. Submitted to IEEE Multimedia Conference, May 1997.

[15] G. Neufeld, D. Makaroff, and N. Hutchinson. Design of a VBR continuous media file server for an ATM network. In *SPIE'96, Multimedia*, 1996.

[16] G. Neufeld, D. Makaroff, and N. Hutchinson. Server based flow control in a distributed continuous media server. In *The 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1996.

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

[18] 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.

[19] 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.

[20] 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`).

[21] 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.

[22] 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.

[23] G. v. Bochmann, B. Kerhervé, A. Hafid, P. Dini, and A. Pons. Architectural design of adaptive multimedia systems. In *Proceedings of IEEE Workshop on Multimedia Software Development*. Berlin, Germany, March 1996.

[24] 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.

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.