Report Number: WUCS-92-26

1992-07-01

Design of a Multimedia Applications Development System

Raman Gopalakrishnan and Andreas D. Bovopoulos

The application envisaged for high speed packet switched networks have diverse and demanding transport requirements that are not satisfied by current communications software implementations. In addition, existing communication interfaces for distributed programs are not general enough to express the communication patterns and control required for distributed applications. This Multimedia Applications Development (MAD) system is designed to address these problems mentioned above. It is intended as a platform for implementing and experimenting with protocols and their implementation, network service access methods, and communication primitives for constructing distributed multimedia applications. It is expected that the research will yield real applications that... Read complete abstract on page 2.

Design of a Multimedia Applications Development System

Raman Gopalakrishnan and Andreas D. Bovopoulos

Complete Abstract:

The application envisaged for high speed packet switched networks have diverse and demanding transport requirements that are not satisfied by current communications software implementations. In addition, existing communication interfaces for distributed programs are not general enough to express the communication patterns and control required for distributed applications. This Multimedia Applications Development (MAD) system is designed to address these problems mentioned above. It is intended as a platform for implementing and experimenting with protocols and their implementation, network service access methods, and communication primitives for constructing distributed multimedia applications. It is expected that the research will yield real applications that can be used with a campus ATM environment, and that the results will provide deeper insights into how the communication needs of applications can be satisfied more effectively.

Design of a Multimedia Applications Development System

Raman Gopalakrishnan and Andreas D. Bovopoulos

WUCS-92-27

July 1992

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

This work was supported by the National Science Foundation under grant NCR-9110183, and an industrial consortium of Bellcore, BNR, DEC, Italtel SIT, NEC, NTT, and SynOptics.

Design of a Multimedia Applications Development System

Raman Gopalakrishnan **Andreas D. Bovopoulos**
Department of Computer Science
and
Computer and Communications Research Center
Washington University, Saint Louis, USA.

Abstract

The applications envisaged for high speed packet switched networks have diverse and demanding transport requirements that are not satisfied by current communication software implementations. In addition, existing communication interfaces for distributed programs are not general enough to express the communication patterns and control required for distributed applications. The Multimedia Applications Development (MAD) system is designed to address these problems mentioned above. It is intended as a platform for implementing and experimenting with protocols and their implementation, network service access methods, and communication primitives for constructing distributed multimedia applications. It is expected that the research will yield real applications that can be used within a campus ATM environment, and that the results will provide deeper insight into how the communication needs of applications can be satisfied more effectively.

1. Introduction

The advent of multiservice networks such as the proposed B-ISDN places demands on protocol and operating system designers to be able to meet precisely the communication requirements of applications. Satisfying these needs is complicated by the wide variations in the transport requirements arising out of the diversity encountered in applications media types and connection modes. Additional complexity results from having to provide application programs, an inter process communication (IPC) abstraction that reflects their

This work was supported by the National Science Foundation under grant NCR-9110183, and an industrial consortium of Bellcore, BNR, DEC, Italtel SIT, NEC, NTT, SynOptics.

Computer and Communications Research Center, Bryan Hall 405 – Campus Box 1115, Washington University, One Brookings Drive, Saint Louis MO. 63130-4899. e-mail: gopal@wuccrc.wustl.edu, andreas@patti.wustl.edu. Fax: 314-935-7302

requirements more accurately, such as associations between multiple endpoints whose configurations and properties may change dynamically [21]. In addition, the nature of the interface to communication services proposed for ATM networks is more varied and complex, and the services more sophisticated than the existing ones [15, 2, 3]. It is therefore necessary to define a framework in which the problems relating to the points raised above can be studied, and proposed solutions can be implemented and evaluated. The aim of this project therefore is to design a software testbed for experimenting with solutions to the abovementioned problems and develop multimedia applications to evaluate their effectiveness. In what follows, we identify the main problems and the issues involved in their solution. We discuss the proposed solutions and the design of the software environment in which they will be implemented. We conclude with a discussion of what we expect to learn from this work and future plans.

2. Main Problems

The following are the main problems that need to be addressed in the current communication scenario.

(A) **The Diversity of Communication Requirements :** There is an increasing number of services that endeavor to offer more than just remote terminal access and file transfer. These services are referred to as *multimedia services* and have communication requirements that range over a wide spectrum. This diversity arises due to the following factors :

1. Applications : The nature of the application determines to a large extent the nature of the transport service required. This means that for a given application, certain transport parameters have stricter constraints than others. For example, a file transfer application would need error free data transfer with high throughput, though it could tolerate larger delays. A remote file system application would require quicker response times in addition, though throughput requirements would be less strict. A real-time audio service could tolerate a certain degree of loss but would be intolerant to a delay that exceeded an acceptable limit.
2. Media : Recent workstations can interface with a variety of input devices and are able to digitally store and process information of several media types. As the computer is increasingly being used for communication rather than just computing, it becomes necessary to address the problem of transporting data of different media types that have incompatible transport and presentation requirements. For example, samples of continuous media such as voice generate constant bit rates and are delay sensitive but error insensitive, whereas hierarchically coded video generates variable bit rate data and is error sensitive.
3. Connection Modes : Current data transport mechanisms typically support two party connections that could be either simplex or duplex. However other connection patterns are becoming necessary [15, 30]. For example, distributional type

services such as for video, require broadcast connections, and conferencing services require multiway connectivity. Multicast is useful for implementing process groups [6] and for name server applications.

The above factors make it necessary for protocol designers and system software developers to include support for these diverse requirements [26]. Since requirements could change and new ones could arise over time, modifiability and extensibility are important. There is considerable amount of uncertainty as to how best this could be achieved.

(B) **Application Abstractions for IPC :** Communication needs of multimedia applications can no longer be met by traditional one-to-one, client-server mode of communication. Application components will have varied patterns of interaction such as multicast sends, multisource input and selective read/write capabilities and rights. The interaction mode may be synchronous or asynchronous with possibly real-time requirements. The main issue of concern here is to be able to determine these characteristics and provide communication primitives that closely model them so that the applications developer can express the application requirements in a concise and flexible manner. In the context of multiple simultaneous media streams, the primitives should allow for specification of temporal relationships that need to be enforced between streams in a generalized media-independent manner [23, 27].

(C) **The Gap Between Transmission Rates and Application Level Throughput :** There is at present a lag between the throughput delivered to an application and the speeds at which data can be transmitted. This implies the existence of a bottleneck [8] caused by a combination of factors such as host architecture (slower buses, high interrupt latency etc.), Operating System (OS) factors (context switch overhead, buffer to buffer copies) as well as protocol factors (processing overheads, choice of error and flow control policies etc.). Also, given that future networks will provide control over the quality of service (QoS) offered to clients, the host software must be able to preserve the negotiated QoS on an end-to-end basis. The solution to this problem requires careful study of the three factors mentioned above and presents a considerable challenge at present.

(D) **The Complexity of Network Service Interfaces :** The future networks will be considerably different from the conventional packet switched WANs (such as the Internet) or the shared access LANs (such as Ethernet). First, they will offer a wide range of services to applications through a standard Application Programming Interface (API) [2, 3]. Further they will also support more sophisticated access mechanisms such as virtual channels with possibly multiple senders and receivers [5], connectionless data transfer services [10], and multiparty calls. In addition, the quality of service provided on the data transfer paths will be negotiable. It is not certain as to how best these services will be deployed and accessed. Providing this service will lead to increased complexity of the User Network Interface (UNI) and the signaling protocols at the UNI.

(E) **The Inherent Complexity of Network Software :** As the range of services offered and applications using these services expands, the task of adding software to support them and maintaining existing software becomes a difficult software engineering issue. There is a strong need to simplify the software architecture and streamline software development to manage the complexity of communication software. This is a serious problem confronting protocol as well as OS implementors.

3. Some Proposed Solutions

As mentioned earlier our objective is to develop an implementation environment for the next generation of multimedia applications. We therefore need to implement and evaluate solutions to the problems mentioned above. The generality of these problems admits a variety of solutions, and it is our aim to incorporate as many of these solutions as possible in our proposed system to be able to evaluate their effectiveness. We begin by describing the issues that need to be considered and some of the approaches that have been taken.

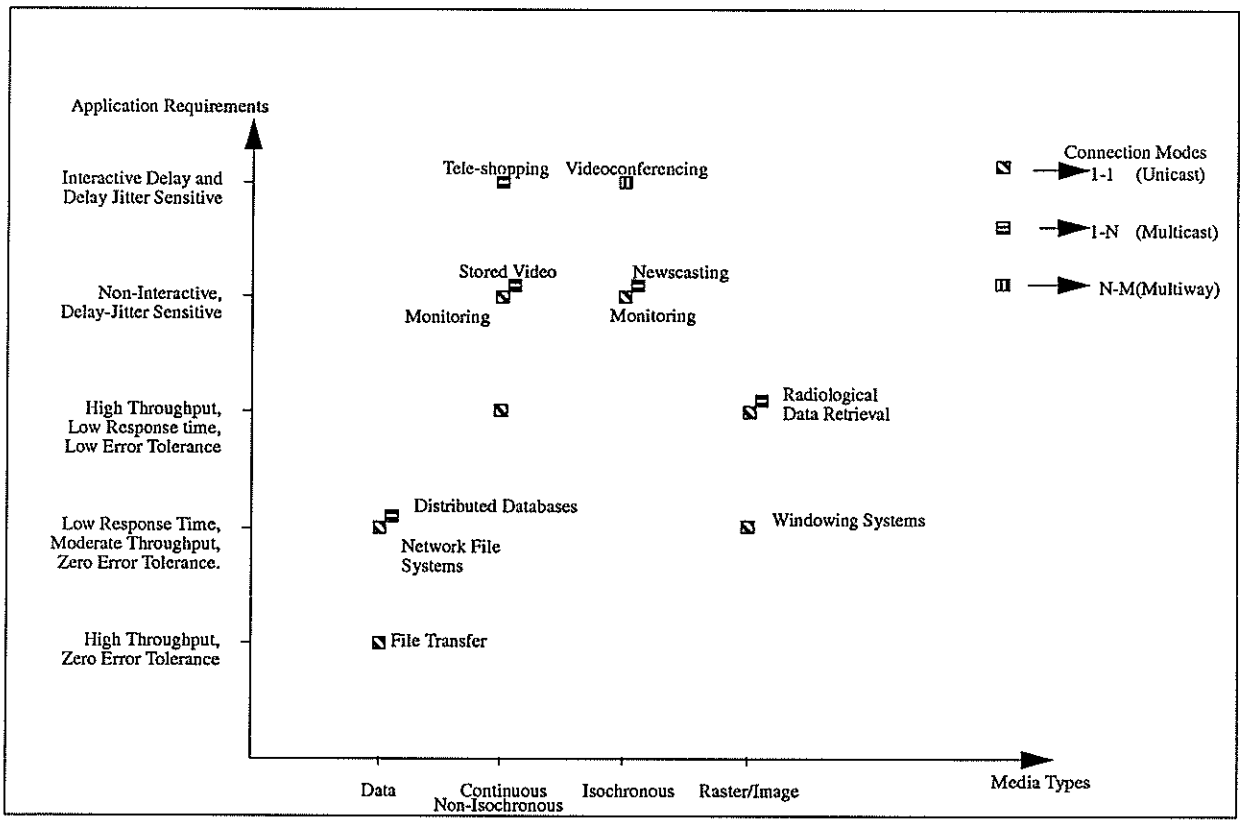

Figure 1: The transport service categories

Handling Diverse Communication Requirements : As discussed earlier the main sources of diversity are : applications, media and connection modes. A possible

approach is to define the space of transport service categories defined by these three factors, and to identify points that represent realistic classes of applications, as shown in Figure 1. Each point represents a Transport Service Category (TSC) and defines a certain mix of requirements that are to be met by the transport subsystem.

For the more common TSCs, protocols that meet their transport requirements have been proposed [13]. The transport system software can be implemented in several ways. One approach is to have a separate protocol hierarchy for each TSC and multiplex the TSCs using a TSC identifier. This approach is somewhat similar to the *protocol domain* concept [20] in BSD UNIX. Another approach would be to have a single protocol providing support for all types, with an initial negotiation phase to determine the transport category to which the application belongs. Another method would be to compose a transport service using predefined modules that implement the appropriate policies for transport functions such as scheduling of transmissions, scheduling of data delivery, acknowledgment generation, error reporting and recovery mechanism, flow control and congestion control mechanism [29]. This approach is more adaptive because as application requirements change during a session, one or more modules could be replaced with functionally equivalent ones or could be eliminated altogether.

Our system incorporates the third approach described above, because we feel that it best meets the needs of future applications. In addition it naturally lends itself to experimentation and fine tuning.

IPC Abstractions for Multimedia Applications : Abstractions like the *connector* and *active devices* [21] together support a programming paradigm for multimedia applications. A multimedia virtual circuit [26] has been proposed to provide temporal synchronization and a coordination mechanism for the conversing parties to easily set up, refer to, modify, and tear down their various media. An *orchestration layer* [26] has been proposed to perform session and presentation functions such as synchronization and provides abstractions for stream sources and sinks. Solutions to the problem of stream synchronization have been dealt with at the application, session and transport layers. Most schemes structure the stream into frames and enforce synchronization in terms of these frames.

Our work provides an IPC abstraction based on an *association* [18]. It implements methods of specifying temporal relations coupled with abstractions for continuous media streams. The synchronized streams are delivered to the application according to the specified order. Again, the idea is to allow the developer to experiment and determine which of the methods suit the application best.

The Processing Bottleneck Problem : It is observed that transmission rates are an order of magnitude higher than the data transfer rates between applications. The processing speed on the host has not increased proportionately compared to transmission speeds, mainly because of the large number of factors that affect host performance and our poor understanding of their interaction. As mentioned above the most important of these factors are host architecture, operating system and protocol processing software.

In typical architectures the CPU, memory and peripheral interfaces are connected over a system bus. Bus latency is avoided by having larger data and instruction caches and wider data paths. This approach works because normal data access is localized and bursty. For high rate real time data, however, this design is inappropriate, and new designs based on a data switch rather than a bus have been proposed. Our work is however independent of these factors and is based upon conventional host architectures.

The Operating System is an area where much attention is focused at present. Current OSs ensure fair sharing of resources and are geared towards feature rich interfaces with not much emphasis on real time response or other performance guarantees. This approach is not suitable for supporting multimedia applications especially since it is important to preserve end-to-end QoS. New resource management techniques that have features of real time Operating systems have been proposed [17] to overcome this shortcoming. Because of their low OS overheads and adaptable design, microkernel operating systems are being considered suitable candidates [19]. Such systems provide only the most rudimentary functions in an efficient manner allowing an application to exercise greater control over resource usage. In addition, these systems are easy to implement and to modify thus facilitating experimentation. In view of our objectives, we adopt the micro kernel approach. Our initial platform will a NeXT machine equipped with an ATM interface.

There are two approaches to improving protocol performance. Current implementations are usually in software and are therefore slower. Proposed solutions are performing protocol processing on silicon [7] or on the network interface card so that variations in system load do not affect protocol performance. Also protocols themselves are being made leaner, given that the transmission technology has become more reliable. Thus, a combination of the two approaches is expected to alleviate the protocol processing bottleneck problem to a large extent. Before an optimal suite of protocols can be obtained several protocol choices must be evaluated in software within the host OS environment. Therefore we also must study the interactions between the implementation of the protocols, and algorithms used for functions such as memory, process and timer management.

The Complex Nature of Network Services : The proposed networks are expected to cater to a wide range of applications with ATM as the single integrated transfer mechanism. They are also expected to offer some processing capabilities within the network (PAL to NTSC conversion for video for example) and other value added services. As a result, the interface to network services will become complex as more services are deployed.

The interface is usually implemented using a signaling protocol at the UNI [24]. A client can communicate with a network controller station using this protocol to request services and respond to network events. The UNI protocol would be complex because of the variety of connection types and their associated services. A modular implementation is therefore desirable so that services could be selectively added and modified. Another approach is to layer the message based signalling interface under a procedural API. This would hide details of the message formats with a portable

and well defined programming interface. Our initial work will use a signaling protocol because of the nature of the existing network environment. An API based approach will be implemented later for some commonly used services.

Software Complexity : Communication software is complex because of the large number of protocols and their conflicting requirements, their strong interaction with the operating system and the need to provide end-to-end transport with real-time constraints and/or at high levels of reliability. Because of the evolutionary nature of requirements and their solutions, a systematic and extensible way of software development has to be adopted so that the complexity can be kept within reasonable limits. Also since the mechanisms to be used are by no means well agreed upon, the system should allow easy modifiability.

These requirements suggest an object oriented software organization and implementation [4, 28]. This allows developers to implement commonly encountered transport system functions in a generic manner and use the inheritance mechanism to implement specific policies. We follow this approach in the design of our system for the reasons stated above.

4. The MAD System

This section introduces the Multimedia Applications Development (MAD) system. The MAD system is intended to be a testbed for development of highly distributed multimedia applications intended for the ATM based campus network at Washington University [9]. It will be used to implement, evaluate and analyze solutions to the problems encountered in a high speed communication environment. It is also intended to study the performance of real applications on the emerging campus network and draw conclusions to guide future work. The distinctive features of this system are a flexible design that allows an application to precisely tailor its transport service parameters, an attempt to bridge the gap between protocol software and transmission rates, a flexible network service programming interface, a set of abstractions at the session layer to support features such as multiparty sessions and temporal synchronization between continuous media streams, and an object library to support application development. An overview of these features is given below.

Flexible Design : As argued earlier, supporting a diverse mix of applications requires a high degree of flexibility and adaptability of the transport system. We achieve this by implementing different algorithms for common transport functions and providing a facility for binding suitable mechanisms for each transport function to a connection. In addition to allowing a choice of the mechanism to perform a function, specific values of parameters can be chosen to precisely meet transport requirements of the connection. These parameters are derived from the QoS specified by the application. Some commonly used combinations along with associated default parameters are organized as TSCs as mentioned earlier and can be used directly in most cases.

IPC Mechanisms : As mentioned earlier the *association* abstraction combines several command and control structures that may be needed to support cooperation amongst

multiple sites, into a uniform and simple abstraction. The dynamic nature of the association accurately reflects communication patterns encountered in typical multimedia applications such as conferencing. The association is primarily intended to provide primitives for managing the sessions between interacting application sites, and managing the transfer of information streams amongst subsets of these sites. Typically, session management involves establishing a session, adding session endpoints, changing default rights, terminating a session and so on. Transfer management operations include setting up streams between session endpoints, providing synchronization of streams, and QoS management.

The Host Bottleneck Problem : The problem of slower hosts depends on several factors some of which (such as hardware architecture) are not within the scope of this research. The issues that we will be concerned with are OS and protocol mechanisms and their interaction. The MAD system has been targeted to be implemented on a microkernel architecture with a high degree of modularity and independence among system components, a low overhead process abstraction such as threads and control over scheduling of threads *. The OS mechanisms that will be implemented are message management, scheduling of responses to asynchronous events (such as protocol events) and synchronous events (such as delivery of video frames), and a message passing mechanism between objects.

Network Service Interface : The design of the interface to network services is important to support the future "intelligent network". Our design explicitly takes this into account and provides an API for network programmers to access network services. The design of the API is influenced to a large extent by the proposed access protocols to the ATM network and the service modes that it offers. The API is used to support network level abstractions such as *Calls* and other network service abstractions with different service characteristics. It provides functions such as addressing, control of call and connection attributes and status queries.

Convenient Prototyping of Applications : The main objective of our effort is to develop a clearer understanding of the application behavior.The MAD system makes it easy to construct applications by creating a library of often used modules and using program developing tools †. The library will allow applications to compose protocol stacks and translate their QoS requirements to suitable parameters of the transport functions. Default configurations that meet most requirements will be provided. The support for application development encourages experimentation and identification of new requirements or improvements in transport system support.

5. MAD System Design and Modules

This section outlines the design of the MAD system architecture. It consists of the following three main components as shown in Fig 2.

*The initial implementation will be on the NeXT machine equipped with an ATM network interface
†The NeXTStep is an example of such a system

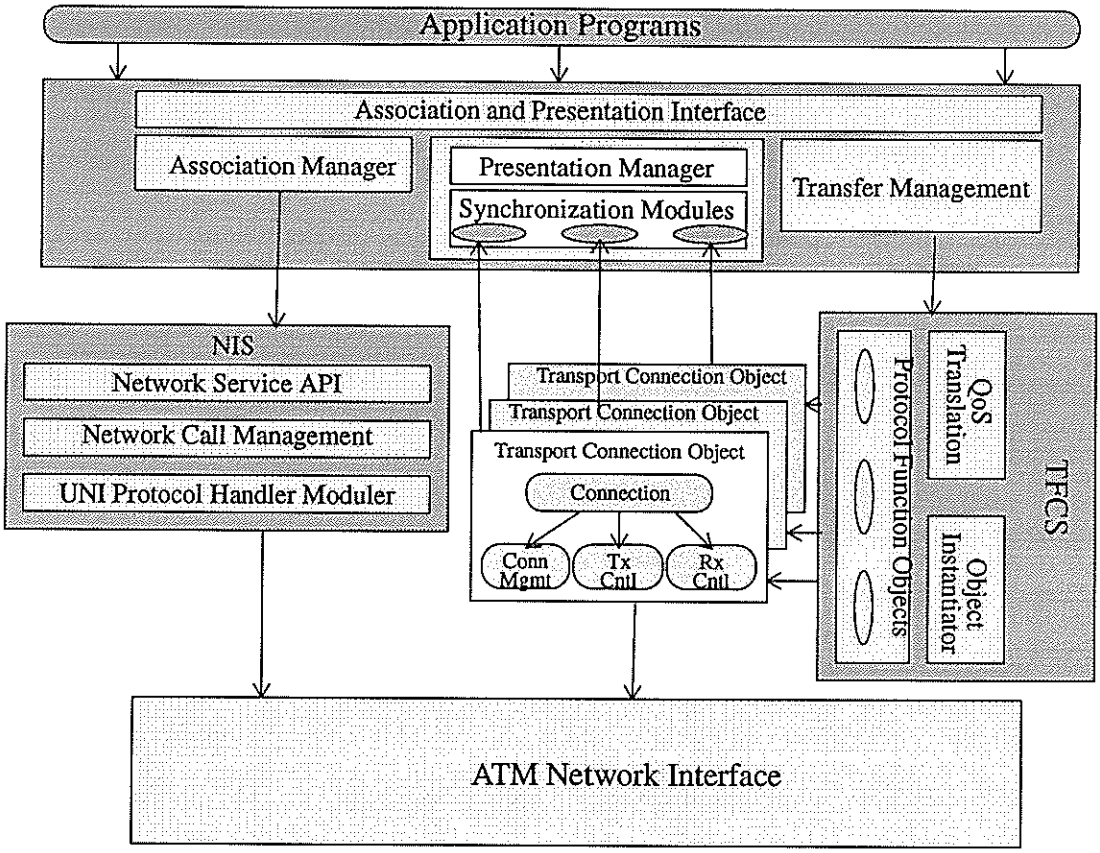

Figure 2: The MAD system architecture

The Network Interface System (NIS) : This module manages all aspects related to accessing network services. It implements the UNI signaling protocol and supports the functions of the API. It supports network addressing, call management and service parameter negotiation.

The Transport Function Composition System (TFCS) : This module encapsulates all transport functions and the protocols that implement them as a set of protocol function objects. It allows an application to compose a protocol stack based on its requirements with appropriate choices of parameter values.

The Association and Presentation System (APS) : This module supports the association abstraction. It allows an application endpoint to create and modify associations, and to specify connections with a desired QoS between the endpoints. It supports primitives for data transfer and session control and maps it to the functions provided by the NIS and TFCS. It also relates data streams carried by the transport system in time and enforces a temporal ordering in their presentation. Usually the information streams involved in a single application will be related in this manner. The APS addresses the problem of specifying, representing and enforcing temporal

relationships in real time and is required in order to support applications such as videoconferencing.

The components are implemented as a collection of objects that communicate using method invocation. Lower level message passing mechanisms are used to access the network interface. The object paradigm is specially suitable for implementing transport functions because it allows the implementation of alternate algorithms while keeping the same interface [22]. In addition the run time facilities provided by the OS can be used to replace one object with another having the same functionality to respond to changing application needs or network status. Each component contains methods to collect run time statistics on parameters specific to that component. The internal structure of these components and the OS support that they are provided is described in greater detail below.

5.1. The Network Interface System (NIS)

NIS is responsible for handling client requests for network services. The main modules of NIS are the API that provides a procedural interface to clients, the call manager that maintains the call and connection status for local clients and a UNI protocol module that implements the signaling protocol between the host and the network. To transfer the UNI PDUs, the NIS uses a suitable transport service provided by the TFCS module. These three aspects of the NIS are elaborated upon below.

Network Service API : The API is a procedural interface that allows a user to gain access to network services. The procedural interface provides transparency from the signaling protocols that are exchanged across the UNI. The functions allow a user to setup a call, supply participant addresses, and setup data connections with suitable transport characteristics. The attributes of the call and subsequent operations vary upon the call model used. The existing model supports the notion of a *call* that binds a set of addresses and allocates network resources among them. A call enfolds a set of *connections* with specific QoS parameters. Therefore the API has procedures to create and manipulate calls, add connections and negotiate their service parameters. The API will be implemented as a user level object that gets instantiated depending on the network service abstraction (eg. Calls) requested by the application.

Call Manager : The call manager maintains state information on the calls that the local node participates in. As the network abstraction for ATM is connection oriented, it is necessary to maintain state information that includes the list of connections per call, the VCI/VPI to be used for transmitting on each connection as well as the VCIs/VPIs that have been assigned to other endpoints of the connections in the call, the transport characteristics per connection, and any other relevant information. The call manager also performs operations on a call in response to API functions invoked from above.

UNI Protocol Module : The ATM network is accessed through a signalling mechanism at the lowest level. The signaling occurs at the network edges also known as the User Network Interface (UNI). The signaling protocol that will be used is called CMAP

[12] or the Connection Management Access Protocol and is to be implemented for the Washington University campus network. The signaling message formats and the request response sequences are used by the call manager to perform the functions requested of it. The signalling messages are carried on a preassigned signalling connection and use a reliable transport protocol to carry them.

5.2. The Transport Function Composition System (TFCS)

The TFCS is responsible for binding a set of transport function objects to a transport connection. The choice of algorithms for protocol functions is guided by the TSC of the connection, or can be explicitly specified by the creator during the establishment phase. The QoS requested is used to guide the choice of arguments (if any) for the objects chosen. These choices also involve negotiation with other endpoints of the connection. The main components of the TFCS are (1) matching TSCs to a set of protocol function objects (PFOs), (2) negotiation of QoS with the network, (3) instantiating the connection and the associated function objects with the negotiated parameters.

Matching Transport Requirements to Mechanisms : This phase is guided by the TSC of the connection and the transport service quality requested. The TSC is specified by the application along with the QoS parameters. The TSC determines the choice of the set of objects that implement various protocol functions. The QoS is used to determine the parameters with which these objects are instantiated. The choice of parameter values is determined by methods defined in the chosen objects. The chosen configuration is then bound to the connection. For example, consider an isochronous (i.e samples generated periodically) voice stream being multicast over a connection. The TSC would dictate a rate based multicast transmission strategy, no acknowledgment generation and no error control. The parameters of relevance would be the size of a transmission unit (the number of bytes in a voice sample), the maximum transmission delay (150 msec), and the delivery rate to the recipient(s) (chosen to be the sampling interval at the source). The choices made above are then negotiated with the NIS as well as the remote peer if the application requires it.

Negotiation with Service Provider : When an application wants to communicate it must establish a call and add connections to it with requisite parameters. This is done by invoking appropriate network services through the API mentioned earlier. The parameters to be passed are the set of endpoint addresses that participate in the call, and a list of connections and their transport characteristics. If network resources are unavailable, then the parameters are either relaxed or a failure status is reported.

The application may also require negotiation at the transport system level. This will be necessary in order to inform the other endpoint(s) of the specific transport algorithms that have been composed for the connection. This negotiation is done on a separate connection between the TFCS entities on each system.

Instantiating the Transport Function Objects : Once the transport protocol mechanisms have been agreed upon and the transport service parameters have been determined, the protocol objects and control path formed by their functional relationships

must be instantiated to perform protocol processing on data exchanged over the connection. These instantiated protocol objects composing the control path constitute a connection and are managed by a connection object. The connection object stores state information and pointers to the objects that support the connection. It also collects statistics and performs overall control of the transport service provided to the connection, such as reconfiguration of connection components.

5.3. The Association and Presentation System

We have seen how the transport requirements of an application's data stream are translated by the TFCS into a transport connection with suitable mechanisms and parameter values. Several such connections are then carried by separate virtual circuits that can provide the bandwidth and delay requirements negotiated for each connection by the transport subsystem. The network in turn supports several access modes such as the call to carry these related virtual circuits used by an application. Since these connections are related by virtue of supporting a single application activity, there must be a mechanism to exercise overall control over the communication activity. This is supported by the APS which provides the association mechanism. The APS has two main components, namely session management and transfer management.

Session Management : This creates the association given a list of endpoint addresses and performs operations such as adding and deleting endpoints, controlling attributes of the association such as accessibility , monitoring and modifiability. Accessibility controls how other endpoints join the association. Monitoring allows an endpoint to be notified of changes in the association attributes.

Transfer Management : This allows an endpoint to setup connections and to specify their QoS. It then allows the creator to negotiate with the transport subsystem to satisfy the QoS requested. Another important function provided is enforcement of temporal relations between data on related connections. It is expected that for continuous and isochronous data, the application will require some sort of temporal ordering to be maintained among the data samples of the streams. A well known example is the "lip-syncing" problem in video communication. Because different virtual circuits have different paths, related streams following different paths lose synchronization within the network and have to be synchronized before they are presented. This task is performed by the APS and is a session layer function. Both temporal as well as event synchronization is supported. There are several ways to specify, represent and enforce temporal relations [27]. The actual mechanism is tied to the representation of continuous media data when it is sent in messages. The message object structure describes the representation in more detail. The synchronization mechanism is implemented by including a synchronization object in the data flow path. Once the temporal relations have been supplied to it, it delivers data to the receiver in a synchronized fashion.

6. Design of Transport Function Classes

As mentioned earlier, the TFCS composes a protocol stack given the application requirements. The basic components of a stack are organized as a hierarchy of classes. These classes are instantiated with parameters (if applicable) and are associated with a transport connection by a transport connection object that stores the state of the connection. The basic functions that are implemented by the classes are described below.

6.1. Transport Function Class Hierarchy

Due to flexibility requirements and the necessity of aiding experimentation, the main transport layer functions have been identified and implemented as separate objects. The granularity of the objects from which a protocol suite is composed is much smaller than the layer by layer composition found in current systems. The objects comprising the transport functions library implement different algorithms for the general transport functions such as connection management, transmission control and reception control. These categories can be further decomposed — for example transmission control, which involves a flow control mechanism (rate based, sliding window,credits, or a combination), an error control mechanism which comprises error detection and recovery (cumulative or selective acknowledgments/retransmissions), and a congestion control mechanism [14] (such as in-call negotiation and cell marking methods for ATM). Figure 3 shows the components of the general transport functions which have been organized in a class hierarchy.

The generic *transport connection* class is a *container class* that encapsulates classes that implement the functions provided by the transport abstractions *TransAbs*, the transmission control *TxControl* and the reception control *RxControl*.

The transport abstraction is the interface that is seen by the user of the transport object. For example a user of a TCP transport object would see a byte stream abstraction and a program that provides digital telephony would use a transport object that provides a real-time, continuous media stream abstraction. Exposing the structure of the information stream to the transport system helps to optimize transport functions to best meet the application needs.

Transmission control is composed of classes that implement the three major mechanisms for error control, flow control and network congestion control. Error control is composed of mechanisms for detecting errors at the sending end (by a timer mechanism or on receiving status information from the receiver — ACK, SACK, NACK etc.), and mechanisms for recovery at the sender by retransmission (go-back-N (GBN), one-at-a-time selective retransmission (OSR) and continuous selective retransmission (CSR) [25]). Flow control mechanisms could be sliding window based (static receiver buffer or dynamic), rate based (static or dynamic) or a combination of both. Congestion control mechanisms would be either static (reservation at peak rate or a combination of peak and average rates) or dynamic (where the methods provide lossless service or best effort service at lower signalling costs).

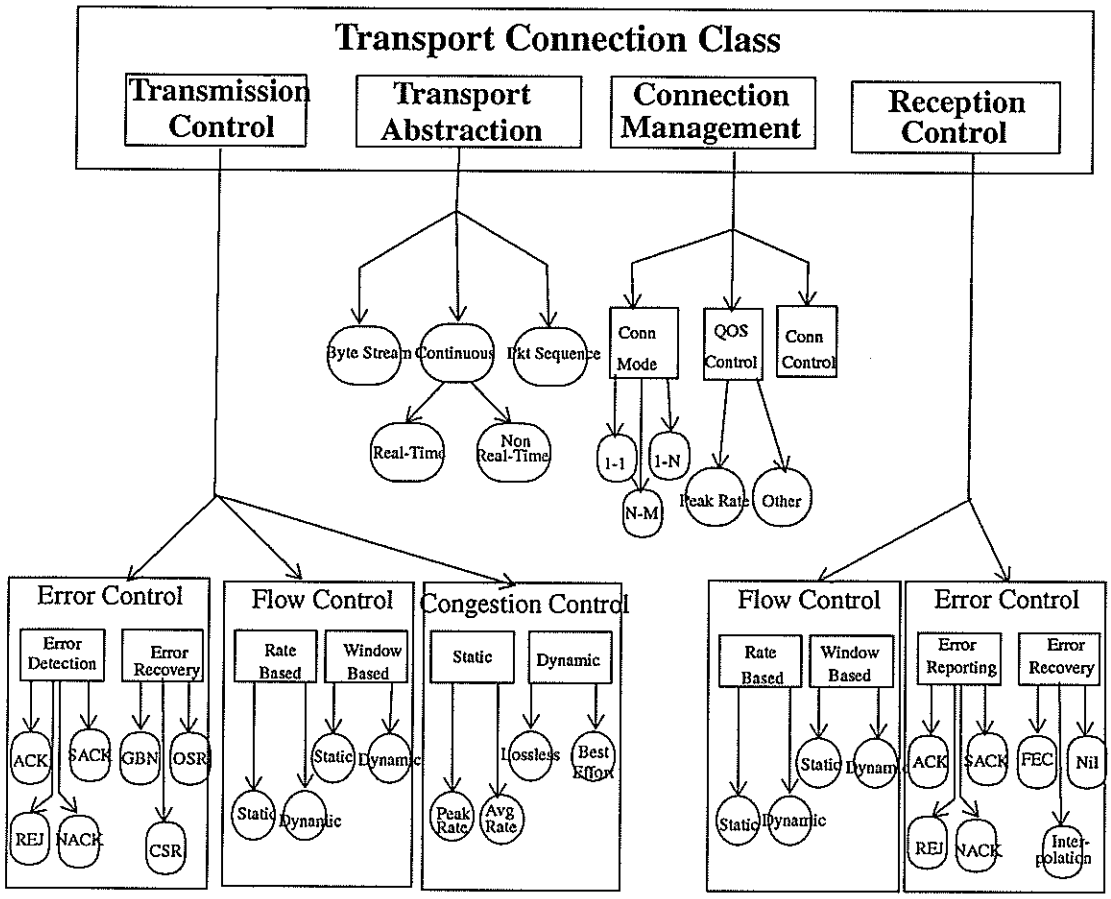

Figure 3: The transport function class hierarchy

Reception control is comprised of classes that implement the complementary functions of the sender. Error control again consists of error reporting mechanisms and local error recovery mechanisms (FEC and interpolation). Flow control mechanisms for window and rate based methods calculate credit/rate based on resource availability and application behavior. Congestion control includes mechanisms to report congestion marked cells to the sender. It also has methods to control data delivery to the application depending on the transport abstraction used.

Each function listed above is implemented by one or more classes of objects that use a proposed algorithm to implement the function. The linkage between these objects is formed by their functional dependencies.

6.2. Transport Connection Class

The state of a connection is maintained by an object derived from the *TConn* class. This class has pointers to the various protocol function objects and the parameters with which they are instantiated. It has methods to exchange message objects with service user objects

and service provider objects. It also has methods to reconfigure the connection parameters depending upon user requests and the state of the network. These objects are instantiated when a connection is setup and exist during its lifetime.

6.3. The Message Object Classes

In addition to protocol function objects described above, message objects are used to carry user and protocol information between the protocol objects. These objects have functions that allow operations on the user data. For example a message object carrying a video sample may have routines for coding and decoding. The message class has subclasses for depending on the medium to which the data belongs. The first level descendants of the general message class are *MsgData* for normal data, and *MsgSamp* for sampled data. The class hierarchy is shown in Figure 4.

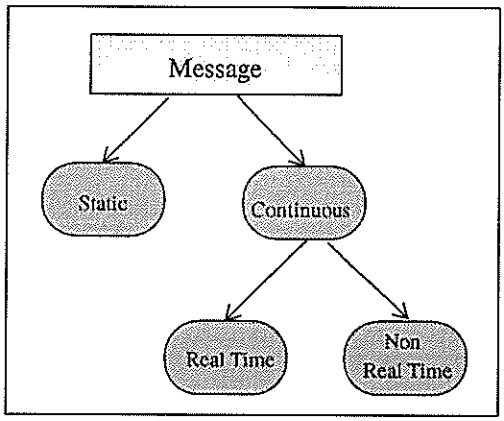

Figure 4: The message object class hierarchy

General functions that operate on message objects such as header manipulation are also provided. The message object mechanism serves to eliminate excessive copying between layers. It is also a more intuitive abstraction for continuous data streams.

7. Future Work

The design of the MAD system reflects the requirements of emerging applications on high speed packet switched networks. We expect that the fine grain composition of transport protocol stacks will help to mitigate the shortcomings of current transport protocol implementations. The flexibility of the system will facilitate experimentation and help determine the best choice of algorithms and transport system parameters to meet application requirements. This will also validate the effectiveness of proposed techniques for various transport protocol functions. The communication interface for applications and the application development environment will be used to develop conferencing and other collaborative applications and test them in the Washington University campus environment. The experience

gained from building the system and using the applications will be used to guide future work in these areas.

References

[1] Arango, M. et. al, "Touring Machine: A Software Platform for Distributed Multimedia Applications", *IFIP*, Vancouver, May 1992.

[2] Arango, M. et. al, "Touring Machine: A Software Infrastructure to Support Multimedia Communications", *4th IEEE COMSOC, International Workshop on Multimedia,* Monterrey, April 1992.

[3] Arango, M. et. al, "The Application Programming Interface to the Touring Machine", Bellcore, Morristown NJ, 1992.

[4] Blakowski, G., "Supporting the Distributed Processing of Multimedia Information in an Object-Oriented, Heterogeneous Environment", *First International Workshop on Network and Operating System Support for Digital Audio and Video*, ICSI, Berkeley, November 90.

[5] Bubenick, R., Gaddis, M.E. and DeHart, J.D., "Communicating with Virtual Paths and Virtual Channels", *Proceedings of the IEEE INFOCOM'92 Conference,* May 6-8, 1992, Florence, Italy.

[6] Cheriton, D.R., "The V Distributed System", *Communications of the ACM,* Vol. 31, No. 3, March 1988.

[7] Chesson, G., "XTP/PE Design Considerations", *Proceedings of the 1st International Workshop on High Speed Networks*, May 1989.

[8] Clark, D.D., "Modularity and Efficiency in Protocol Implementation", RFC 817.

[9] Cox, J. R. and Turner, J., "Project Zeus: Design of a Broadband Network and its Application on a University Campus," Washington University, Department of Computer Science, Technical Report WUCS-91-45, 1991.

[10] Crowcroft, J. and Wakeman, I., "A Technique for the Transmission of IP Datagrams Over B-ISDN Networks", Draft RFC, IETF, April 1992.

[11] Crutcher, L. and Grinham, J., "Connection Management for an Integrated-services Network and its Application to the Multi-media Communications of a Distributed Team", *First International Workshop on Network and Operating System Support for Digital Audio and Video,* ICSI, Berkeley, November 1990.

[12] DeHart, J.D., Gaddis, M.E. and Bubenik, R., "Connection Management Access Protocol (CMAP) Specification", Washington University, Department of Computer Science, Technical Report WUCS-92-01, February 1992.

[13] Doeringer et. al, "A survey of Light Weight Transport Protocols for High Speed Networks", *IEEE Transactions in Communication,* Vol. 38, No. 11, November 1990.

[14] Doshi, B.T. and Johri, P.K., "Communication Protocols For High Speed Packet Networks", *Computer Networks and ISDN Systems,* Vol. 24, 1992.

[15] Gaddis, M.E, Bubenick, R. and DeHart, J.D., "A Call Model for Multipoint Communication in Switched Networks", *Proceedings of ICC,* 1992.

[16] Gopal, G., Herman, G. and Vecchi, M.P., "The Touring Machine Project: Toward a Public Network Platform for Multimedia Applications", *Proceedings of SETSS,* Florence, March 1992.

[17] Hanko, J. et. al "Workstation Support for Time-Critical Applications", *2nd International Workshop on Support for Digital Audio and Video,* Heidelberg, November 1991.

[18] Hong, Z. and McCoy, W., "An Associated Object Model for Distributed Systems", *Operating Systems Review,* October 1990.

[19] Hutchinson, N.C. and Peterson, L.L., "The x-kernel: An Architecture for Implementing Network Protocols", *IEEE Transactions on Software Engineering,* January 1991.

[20] Leffler, S., et. al, *The Design and Implementation of the 4.3BSD UNIX Operating System,* Addison-Wesley, 1989.

[21] Leung, W.H. et. al, "A software architecture for workstations supporting multimedia conferencing in packet switching networks", *IEEE Journal on Selected Areas in Communications,* April 1990.

[22] Lippman, S.B., *The C++ Primer,* Addison-Wesley, 1991.

[23] Little, T.D.C. and Ghafoor, A., "Multimedia Synchronization Protocols", *IEEE Journal on Selected Areas in Communications,* December 1991.

[24] Minzer, S. and Spears, "New Directions in Signaling for Broadband ISDN", *IEEE Communications Magazine,* February 1989.

[25] Netravali, A.N., Roome, W.D. and Sabnani, K.K., "Design and Implementation of a High Speed Protocol", *IEEE Transactions on Communications,* Vol. 38, pp. 2010-2024, 1990.

[26] Nicolaou, C., "An Architecture for Real-Time Multimedia Communications Systems", *IEEE Journal on Selected Areas in Communications,* April 1990.

[27] Ravindran, K., "Real-Time Synchronization of Multimedia Data Streams in High Speed Networks", Technical Report, Kansas State University, January 1992.

[28] Schill, A., "Objects and Distribution: Advantages, Problems and Solutions", *Intelligent Tools Conference,* Paris, November 1989.

[29] Schmidt, D.C., Box, D.F. and Suda, T., "ADAPTIVE A Flexible and Adaptive Transport System Architecture to Support Multimedia Applications on High-Speed Networks", Technical Report 92-46, Department of Information and Computer Science, University of California, Irvine, 1992.

[30] Yavatkar, R., "Communication Support for Collaborative Multimedia Applications", Technical Report 181-91, Department of Computer Science, University of Kentucky, 1991.