DRAFT v1.1

MULTIMEDIA SERVICE MANAGEMENT AND CONTROL ON ASYNCHRONOUS TRANSFER MODE NETWORKS IN-HOUSE PROJECT - INTERIM TECHNICAL REPORT

Bradley J. Harnish
Distributed Information Systems Branch

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

Table of Contents

List of Figures

1.0 Introduction

This Technical Memo serves as an interim report for the Multimedia Service Management and Control on Asynchronous Transfer Mode (ATM) Networks In-house project. It documents the ideas that have been addressed so far, describes the results of their application, and describes the direction of the remaining effort of the project.

2.0 Background

In 1995 we started an Exploratory Development (6.2) project with Columbia University entitled ATM Management and Control Application Programming Interfaces (APIs). The effort developed and demonstrated generic software interfaces between distributed computing applications and the distributed applications that manage and control the network infrastructure. The interfaces enabled cooperation between the applications and the network to improve the performance of the overall information systems.

At the completion of that project, the Multimedia Service Management and Control on ATM Networks In-house project was created under the same 6.2 funding (PROJ4519) to experiment with the proof-of-concept prototypes produced by the Columbia University effort in light of Air Force requirements. About the same time, the principal investigator of the Columbia University effort started Xbind, Inc to further develop the prototypes into a commercial product line. Within a year, we signed a Cooperative Research and Development Agreement with Xbind, Inc. to experiment with their commercial products, and this in-house project was re-directed to support our portion of that agreement.

The ATM Management and Control APIs project built on the concepts and early prototypes that originated from the Control, Management, and Telemedia Group of the Center for Telecommunications Research, which was created at Columbia University by a grant from the National Science Foundation. We funded the development of the ideas promoted in Columbia's proposal and successfully demonstrated a proof-of-concept system based on the BSD Unix

platform. Xbind, Inc. re-designed the prototypes for the Windows NT™ platform and developed various drivers for commercial networking and multimedia peripherals.

This report is organized as follows. Section 3 summarizes the concepts developed under the ATM Management and Control APIs project and their manifestations in the IEEE P1520 and ISO/IEC MPEG-4 standards. Section 4 provides brief descriptions of xbindIP and xbindTV, Xbind Inc.'s proprietary implementation of the MPEG-4 Delivery Multimedia Integration Framework and a digital video application that uses it, respectively. Section 5 describes a distance learning system prototype that was developed by the author based on the Xbind products. Section 6 describes the laboratory testbed used for this project. Section 7 summarizes future work, and conclusions are presented in Section 8.

3.0 Underlying Concepts

This section of the report summarizes the underlying concepts of this effort that were developed under the ATM Management and Control APIs project, a 6.2 effort with Columbia University. The effort enabled cooperation between distributed computing applications and the distributed software that manages and controls the networks on which they rely. The distributed computing applications control the networking resources they need by interacting directly with objects in the network control software that dynamically provision, maintain, and de-allocate ATM network resources based on the application's specific requirements. The Final Technical Report was published as AFRL-IF-RS-1999-54.

This approach involved the application of distributed object computing technology to the network management and control software and provided generic application programming interfaces (APIs) through which the distributed computing applications could dynamically influence the network state in a controlled manner. Middleware applied a level of abstraction between the distributed computing applications and the network systems in such a way that the applications could request generic services from the middleware without requiring specific knowledge of the underlying network technology. The middleware provided the abstract services to the applications by implementing the system calls on the underlying systems.

The concepts developed in the ATM Management and Control API effort stem from the Extended Reference Model (XRM) shown in Figure 1.

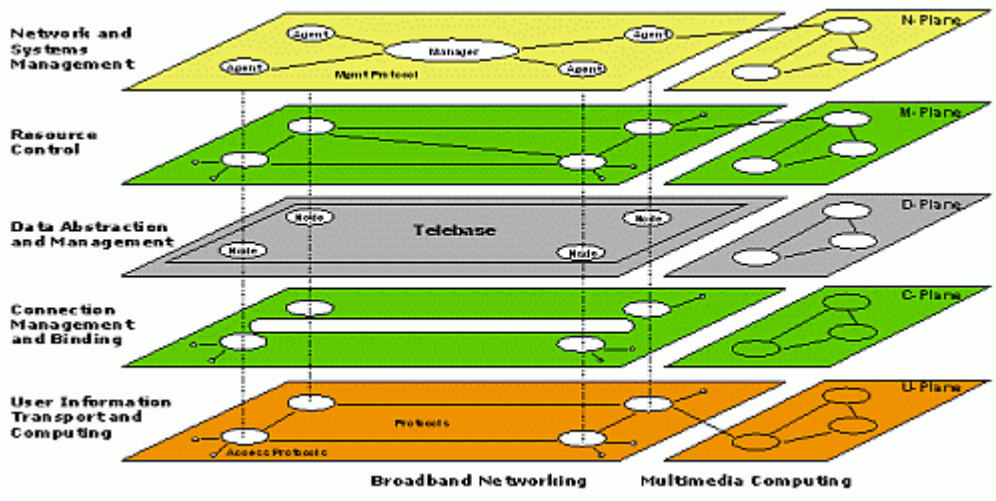

Figure 1. The Extended Reference Model (XRM)

The XRM illustrates a functional decomposition of a distributed object system environment made up of broadband networking and multimedia computing resources. The system state is represented in the Data Abstraction and Management plane, or Telebase, which is distributed throughout the system. Algorithms in the remaining planes monitor and/or manipulate the Telebase to provide the appropriate functionality. One of the fundamental characteristics of the XRM is the separation of transport and control. Algorithms for information transport are represented in the bottom plane, while algorithms for network and end system control are represented in the two planes adjacent to the Telebase plane.

Using an example to illustrate the XRM, an object in the Connection Management and Binding plane might be instantiated by an application to create a virtual circuit in the broadband network with specific performance characteristics. It could first interact with routing/location, namespace, capacity and admission policy objects in the Telebase to gather the intelligence needed to perform the task on behalf of the application. It could then interact with objects representing the appropriate switches and network interface cards of the selected path to set parameters and invoke methods to bind the resources of the end-to-end connection. These objects would, in turn, invoke methods in the physical equipment they control to set up the virtual circuit. The

application could then use the virtual circuit object, a protocol stack object, and end system device objects in the User Information Transport and Computing plane to transport data across the network.

The routing tables in this example could be maintained by objects in the Resource Control plane using policies set by objects in the Network and System Management plane. Interaction between objects is through well defined interfaces and remote invocation mechanisms defined by the specific distributed object framework, such as Common Object Request Broker Architecture or Java. Interoperability between objects in this model relies on standard interfaces rather than standard algorithms or protocols.

The ATM Management and Control API project demonstrated the viability of these concepts and demonstrated the need to formally define open standard interfaces between cooperating and competing entities in a distributed object environment. The Institute of Electrical and Electronic Engineers **(IEEE) Proposed Standard for Application Programming Interfaces for Networks (**P1520**)** was initiated shortly after the completion of the ATM Management and Control APIs project to do just that. In contrast to standardizing algorithms and/or protocols that reside within layers of other reference models, P1520 focused on standardizing the control interfaces between layers. The P1520 Reference Model is shown in Figure 2.

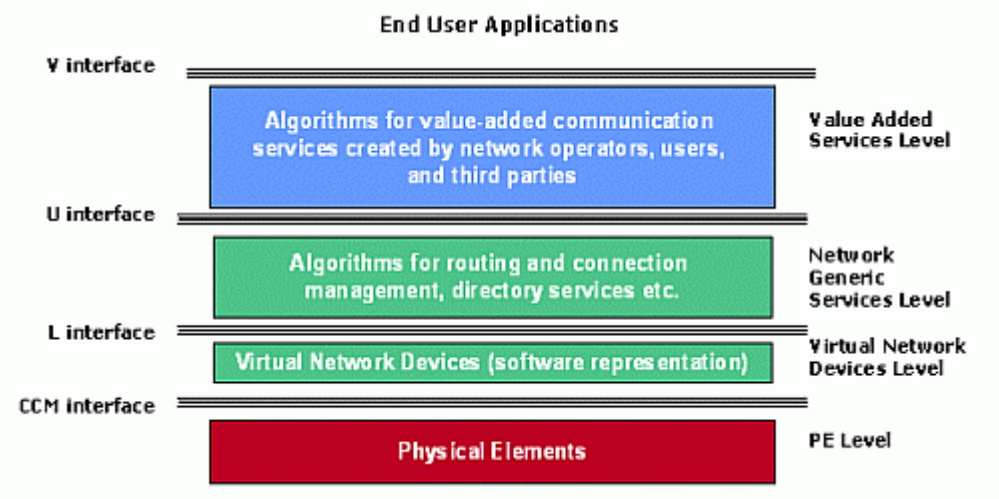

Figure 2. The IEEE P1520 Reference Model

Physical elements of the broadband network and multimedia computing devices, represented in the PE Level, are abstracted in the Virtual Network Devices Level. Algorithms in the Network Generic Services Level operate on these abstractions to provide a set of basic services that are then composed by algorithms in the Value Added Services Level to provide more complex services to End User Applications.

As mentioned above, the focus of P1520 is on standardizing the various control interfaces, labeled on the left side of Figure 2, used for inter-level interactions. In this model, equipment vendors provide the Configuration, Control, and Management (CCM) interface as part of their product offering to allow P1520-compliant software to control them. Best-of-breed competition is enabled at individual or multiple levels of the Reference Model to meet end-user requirements, while allowing the implementations of physical elements, virtual devices, and the management and control algorithms to evolve over time as new technologies are developed. The layered abstractions and standardized interfaces insulate the components in each level from the detailed implementations of components in adjoining levels to sustain interoperability over time.

Another important development since the completion of our effort with Columbia University was the release of the International Standards Organization Moving Picture Experts Group MPEG-4 standard [ISO/IEC 14496-6] which defines the **Delivery Multimedia Integration Framework (DMIF) and the DMIF Application Interface (DAI)**.

The MPEG-4 standard defines an object oriented audio/video encoding scheme that specifies how individual components of an audio/video program are spatially and temporally related during the encoding and decoding of a multimedia presentation. Elementary streams of data, audio, and video are treated as objects in the encoding and decoding process, and mechanisms are defined to describe how they interrelate in time and in space for presentation on the user display.

If one considers a television news program as an example, the program would start with an initial flurry of animated graphics and lead into an anchor person in a studio presenting the news. At the bottom of the screen may be one or more streams of text showing the latest prices of stocks being

traded at the major stock exchanges, a display of the current time and temperature, and the news company's logo. Another area on the screen may be dedicated to the textual display of the current news headlines unrelated to the story being presented by the anchor person. Over time, the video may switch to a live report or file footage related to an ongoing news story. In some cases, there may also be sub-titles displayed or separate audio channels available for a multi-lingual audience. If the program was produced for an interactive environment such as the world-wide web or an information kiosk, there might be objects the user can select to view related programs, download related information, participate in surveys, and purchase advertisers' products.

In the MPEG-4 scheme, each of these elements of the program could be represented by a separate elementary stream. The video of the anchor person in the studio may be sent as a single video stream, or the video stream of the anchor person might be superimposed at the destination onto a synthetic graphical representation of the studio sent as a separate stream. The stock ticker, time, temperature, headlines, and sub-titles could all be separate textual data streams, but they would vary at different rates and have different timing constraints for synchronization with the other elements on the screen. The various video and audio streams would have higher bandwidth requirements and more stringent timing constraints than the textual data streams. As a result, there could be dozens of elementary streams delivered to create the overall program, all requiring a different quality of service from the transport medium.

To maximize the flexibility of the standard, the MPEG-4 Committee defined the Delivery Multimedia Integration Framework (DMIF) as part of the MPEG-4 standard. It is a framework that separates the encoding/decoding and synchronization tasks of a multimedia application from the task of delivering the multimedia streams. Further, it defines a session layer protocol that manages and controls multimedia streams over an abstract transport infrastructure that can be implemented using broadcast technology, interactive network technology, and/or disk technology.

Figure 3 shows a point-to-point case of the framework for interactive network transport. The distributed application shown at the top of the diagram is responsible for the encoding, decoding,

and synchronization of the various elementary streams that comprise the interactive multimedia program. The DMIF system, shown at the bottom of the diagram, is responsible for delivering elementary streams while remaining unaware of any synchronization between them. Therefore, the DMIF system can be used as an all-purpose delivery tool for many types of streaming data. It is not limited to providing support for MPEG-4 applications

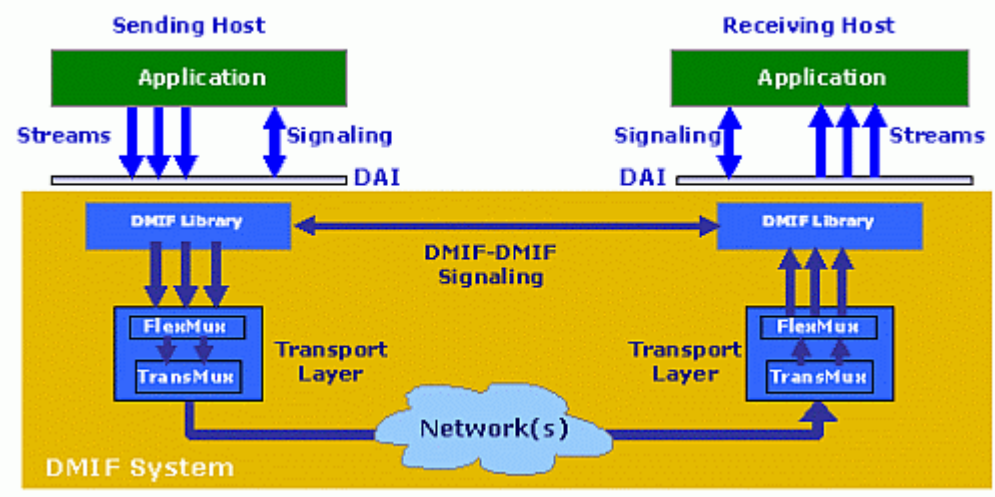

Figure 3 –The MPEG-4 Delivery Multimedia Integration Framework

Communication between the application and the DMIF system is defined by the DMIF Application Interface (DAI). It provides a small set of transport-independent primitives that handle signaling (i.e. management and control messages) separately from the transport of elementary streams, as indicated in Figure 3. The management and control primitives of the DAI enable the establishment of sessions between the application and the DMIF system (DAI_Service_Attach and DAI_Service_Detatch) and the management and control of multiple channels within those sessions (DAI_Channel_Add and DAI_Channel_Delete). The DAI's stream transport primitives allow the application to send and receive byte stream data (DAI_Send_Data and DAI_Receive_Data) and application control data (piggybacked onto service and channel primitives) to and from the DMIF system.

Relating to the DAI development effort, one of the Committee members said, *"A significant part of the design effort [was] devoted to the identification of the parameters that should be exposed at this interface, focusing the study on the semantic value that such parameters should carry"*.

[Franceschini, “The Delivery Layer in MPEG-4” *Signal Processing: Image Communication* Volume: 15, Issue: 4, January, 2000, pp. 347-363].

The establishment of channels through the DMIF system includes parameters that allow the application to specify the Quality of Service (QoS) it expects the DMIF system to provide. By establishing the requested channel over a QoS enabled network for the application, the DMIF system agrees to provide the specified QoS. The application is concerned with the QoS of the channel, but is unaware of the transport technology used by the DMIF system to provide it. All of the DAI primitives were designed to hide transport related details from the application, while ensuring end-to-end interoperability between applications.

The DMIF system multiplexes the channels established by the application into a smaller number of channels for transport across the network resources it controls, which could include a combination of network resources that implement a wide variety of transport services with varying degrees of QoS. As shown in Figure 3, the application sends its elementary streams through the DAI over FlexMux channels it has established through the DMIF system. These are multiplexed according to similar QoS requirements by the DMIF system onto a fewer number of TransMux channels it has established with various network transport service providers. TransMux channels are outside the scope of the MPEG-4 standard and are left as implementation details for DMIF system service providers.

The DMIF-to-DMIF signaling channel shown in Figure 3 (which includes the DMIF Network Interface) ensures end-to-end interoperability over a wide variety of network transport technologies. The DMIF system is responsible for mapping DAI signaling messages into the native signaling messages required by the transport technologies it employs.

DMIF and the DAI provide application developers with the flexibility to ensure that their multimedia content is delivered in the most appropriate manner as specified by their applications. At the same time, it insulates them from the details of the underlying transport technology. This allows their applications to function as designed even if the transport technologies change over time, as they probably will. The DMIF system is responsible for adapting to the features and/or

limitations of new transport technologies to ensure that the DAI services are maintained. At the same time, the abstraction of the streams presented to the DMIF system and the session protocol embodied in the DMIF Network Interface, allow the DMIF system to support applications that are not using MPEG-4 coding and synchronization.

4.0 XbindIP^{TM} and XbindTV^{TM}

Through a Cooperative Research and Development Agreement with Xbind, Inc., AFRL/IFGA has installed an implementation of the MPEG-4 Delivery Multimedia Integration Framework (DMIF) system. The installation provides an experimental platform for the development of multimedia applications that use the DMIF Application Interface. The version of the DMIF system we have currently provides transport services with varying Quality of Service (QoS) guarantees to multimedia applications over Internet Protocol (IP) and Asynchronous Transfer Mode (ATM) infrastructures. The reference implementation of the DMIF system is embodied in an Xbind, Inc. product called xbindIP^{TM}. We have also installed a video distribution application called xbindTV^{TM} that uses the DMIF Application Interface to derive services from xbindIP. This section of the report provides a brief description of both of these products.

The Xbind, Inc. product literature at http://www.xbind.com/solutions.htm states that, "***xbindIP^{TM}*** *creates on-demand multimedia services by configuring and programming the three main attributes of any network based multimedia service: connectivity, bandwidth, and QoS. Connectivity allows the interconnection of several peers to create a networked community. For instance, point to point (e.g. unicast), point to multi-point (e.g., multicast), multipoint to multipoint (e.g. virtual network), point to many points (e.g. broadcast), etc. Bandwidth provides the necessary bit rate to deliver multimedia content among peers belonging to the community. In other words, the bandwidth is the thickness or diameter of the pipe that carries information. QoS provides the basis for delivery of rich multimedia content against raised community expectations. The combination of delay and loss bounds defines the "color" of the service. QoS is the delivered guarantee of bandwidth.*"

Figure 4 shows the system architecture of the xbindIP implementation of the DMIF system, which incorporates unique programmable transport capabilities. [from Huard, Lazar, Lim, and Tselikis, “Realizing the MPEG-4 Multimedia Delivery Framework”, IEEE Network, November/December 1998].

The separation of transport and control can be seen from the two groups of components used to process the DAI primitives. The DAI’s data primitives are processed by the user (U) plane components on the circumference of the diagram, while the DAI’s service and channel primitives are processed in the control (C) plane components that occupy the interior of the diagram. The components in the data path along the bottom of Figure 4 are consumer/producer engine (CPE) components which are controlled by corresponding consumer/producer front end (CPFE) components in the middle of the figure. The CPEs process and transport the data under the dynamic control of the CPFEs, which are themselves dynamically controlled by the protocol stack builder (PSB) component. The CPFEs interact with the CFEs through a QoS-enabled Engine Control Interface (qECI) as depicted in Figure 4.

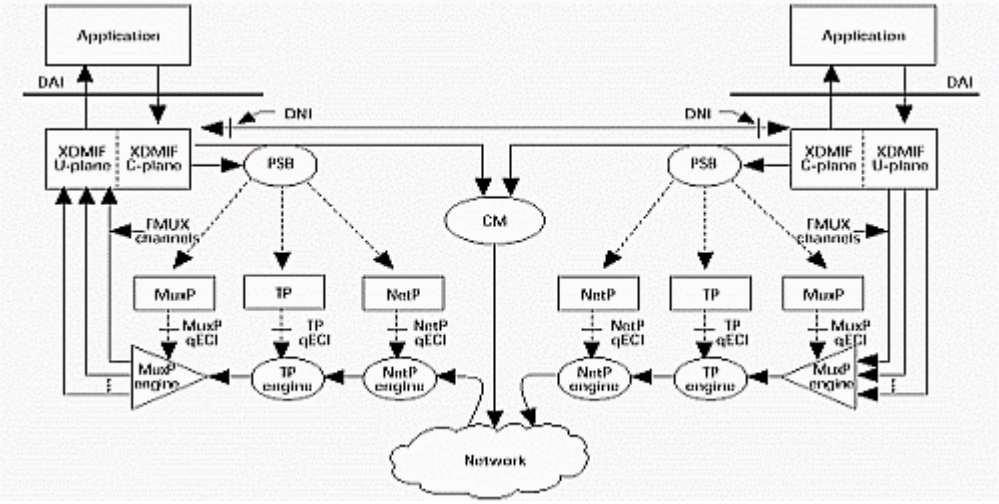

Figure 4 - Xbind implementation of DMIF with programmable transport

The multiplexer port components (MuxP and MuxP engine) multiplex the end-to-end FlexMux channels onto a TransMux channel through the network. The transport protocol components (TP

and TP engine) provide end-to-end communication capabilities through the network, while the network provisioning components (NetP and NetP engine) provide network and data link layer capabilities. The PSB chooses the appropriate protocol stack (combinations of instantiated MuxP, TP, and NetP components) based on parameters used by the application in the DAI channel primitives to setup the FlexMux channels.

Multiple TransMux channels can be used to provide different levels of QoS to the application based on the QoS specified for each FlexMux channel. Examples include TCP/IP, UDP/IP, RTP/UDP/IP, and AAL5/ATM. The PSB is responsible for multiplexing FlexMux channels onto each TransMux channel and dynamically provisioning the necessary protocol stack to support the specified level of QoS. Multiple instances of a TransMux channel could each be supported by a protocol stack that itself could be instantiated with different parameters, as specified by the application through the DAI. When connection oriented networks are used by the DMIF system, the PSB relies on the connection manager (CM) component to setup the required connections through the network.

Relating the components in Figure 4 to the IEEE P1520 Reference Model shown in Figure 2, the CPEs are members of the Virtual Network Devices Layer, the CFPE, PSB, and CM components would be members of the Network Generic Services Layer, and the C-plane and U-plane DMIF library components would reside in the Value Added Services Layer.

xbindTV™ is a video peering application that use the DAI to derive DMIF services from xbindIP. It provides users with point-to-point, multicast, and broadcast distribution of full-screen broadcast quality video on WindowsNT™ personal computers. The application has two parts, xbindTV Player and xbindTV Station, that operate as client and server, respectively. The version that we obtained through our Cooperative Research and Development Agreement supports the distribution of streaming MPEG-1 and MPEG-2 video over Internet Protocol (IP) and Asynchronous Transfer Mode (ATM) networks. Figure 5 shows the point-to-point example of the peering relationship.

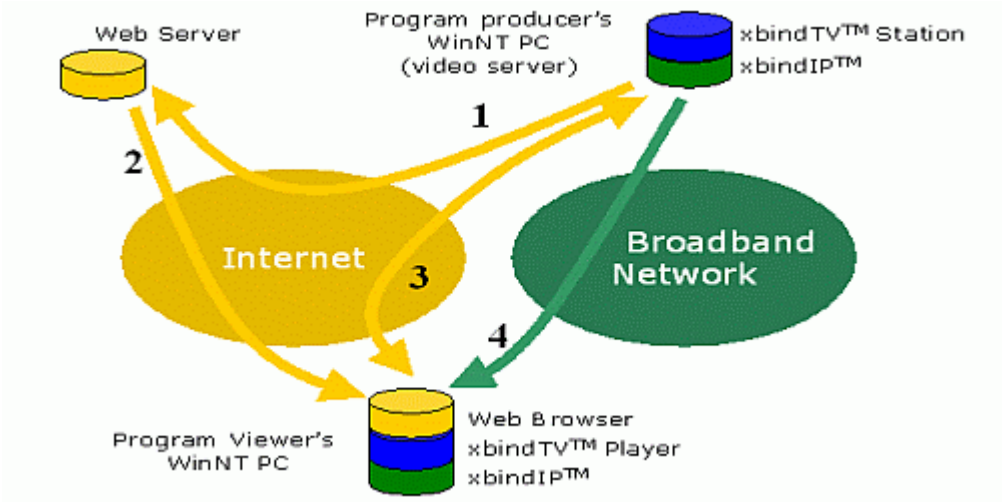

Figure 5 - Point-to-point example of xbindTV

The xbindTV Station application supports the producer of a multimedia program by controlling the MPEG encoder at his or her desktop and orchestrates the flow of the MPEG stream through the DAI. It also allows the producer to specify various parameters about the program, including its format (MPEG-1 or MPEG-2), signaling and transport channels (ATM or IP), a textual description of the program content, and the webserver where this information is to be published. The information is sent to the web server, as shown by arrow #1 in Figure 5.

The xbindTV Player program controls the MPEG decoder at the viewer's desktop and orchestrates the flow of the program stream from the DAI. A prospective viewer of multimedia programs would go to the appropriate website and browse through descriptions of available programs. If they select the link to an xbindTV program, the URL to the associated xbindTV Station is sent through the Internet to the viewer through his or her web browser, as depicted by arrow #2 in the diagram. The xbindTV Player application is started by the browser and receives the URL of the appropriate xbindTV Station. It then uses the DAI implemented by xbindIP to contact the xbindTV Station application and set up the necessary connections through the appropriate network, as described above and shown by arrow #3. Assuming the Station and Player agree to use a Broadband Network to which they each have access, the program content then flows from the Station to the Player, as shown by arrow #4. If the selected program is being

delivered as a multicast, the viewer would be added to an existing multicast session and the connection represented by arrow #4 would be added to the multicast tree for network routing.

The combination of xbindIP™ and xbindTV™ provides a commercial off-the-shelf solution for the distribution of digital video between networked personal computers with the appropriate codecs. No preset network topology is assumed, but the delivered QoS will depend on the capabilities of the network connection(s). The versions of xbindIP and xbindTV provided to us under our CRADA support MPEG-1 and MPEG-2 streams over ATM and IP networks. In this case, ATM connections to the desktop PC must be made through specific ATM network interface cards (NICs) and switches which can be controlled by xbindIP. The IP service is strictly best effort delivery. More recent versions also support multi-resolution RealMedia™ streams and players and network connections through specific Digital Subscriber Loop Access Multiplexor (DSLAMs) and IP routers. Customized user configurations are also supported.

5.0 A Distance Learning Application Prototype

As described above, the combination of xbindIP and xbindTV provides the ability to distribute high quality video simultaneously to several locations with inexpensive commercial off-the-shelf personal computer peripherals. The combination can be used as a foundation for a class of applications where an audio/video program stream is distributed to a large number of viewers, such as a virtual classroom, public/staff address, virtual shareholders' meeting, cultural performance, etc. For the sake of brevity, this type of application is called a distance learning application in the discussion below.

XbindIP and XbindTV enable the interconnection of several peers to create a networked community, or a virtual classroom, of geographically dispersed instructors and students. The students could each be at a separate location, gathered into a number of classrooms or auditoriums, or both. The digital video could be displayed on the student's computer screen or projected onto a classroom wall or movie screen in an auditorium. The availability of course content can be published on a web page with hyperlinks that allow each participating

student/teaching assistant to initiate the distribution of the instructor's live streaming digital video lecture with the click of a mouse.

With a slight modification to the program, the course availability information entered by the instructor with the XbindTV Station application could be augmented to include additional links for course notes, homework assignments, and other related materials (e.g. instructor biography, policies, related papers, product information, annual reports, instructor evaluation surveys, scripts for on-line exams, etc.).

One of the necessary distance learning components missing from the xbindIP/xbindTV combination is a feedback channel that permits the students to ask the instructor questions. There are several possible solutions to providing this capability, some of which are described below.

The **public switched telephone network** could provide this feedback channel fairly well. Assuming the students are not connecting to the virtual classroom with a dial-up modem (because the available bandwidth is not conducive to high quality video), the instructor could announce the beginning of the question and answer period of the class and show the appropriate telephone number for the students to call. As the calls come in, the telephone conversation could be projected onto the outgoing audio channel of the live streaming video with a simple speaker phone located near the instructor's microphone. The instructor would then provide the answer in front of the camera for the entire class to hear.

Electronic mail could also provide the feedback channel by allowing students to e-mail their questions to the instructor. A teaching assistant could screen the incoming questions before passing them on to the instructor who would summarize the question and provide an answer as part of the streaming video distributed to the rest of the class. Less desirably, the instructor could answer the questions outside of the class by responding to each e-mail individually, or answering them collectively and aggregating the questions and answers into a single e-mail sent to each student or into a document published on the course description web page.

A **chat room** program could also be used to allow students to ask questions by typing them into the chat room program which could be started by clicking on a link on the course description

web page. The questions would be displayed on the screens of the instructor and each student that chooses to participate. However, the instructor could easily lose control of the question and answer session without controls that regulate the orderly display of questions and answers.

The author chose to address this problem by developing a Java application that provides an on-line feedback channel to the instructor and allows the instructor to control the question and answer session. It is called the **Question and Answer Service** (QAService), and combines the beneficial features of the e-mail and chat room approaches mentioned above. The QAService consists of two applications called QAInstructor and QAStudent. Each is described below.

QAInstructor runs on the instructor's host computer and is started by the instructor. The URL to the QAInstructor would presumably be included on the course description web page that is published by the instructor's xbindTV Station application. When a student selects the QAService link on the web page, the URL to the QAInstructor is downloaded to the student's computer through his or her web browser. As a result, the QAStudent application is started on the student's computer.

QAInstructor maintains a list of all registered users, a list of students waiting to ask a question (which is a subset of all registered users), and a list of questions submitted by the students. It also maintains a display that allows the instructor to see how many students are registered and how many students are waiting to ask a question. The display has controls that allow the instructor to view each submitted question sequentially and update the display accordingly. The question should be read and answered aloud by the instructor over the live multimedia stream being controlled by the xbindTV applications. QAInstructor receives control messages from all the registered QAStudent applications, performs the appropriate action based on the content of the control messages, and, if necessary, sends out messages to all registered QAStudent applications to update their displays, which are described below.

QAInstructor's initial display is shown in Figure 6. The # of Registered Users counter keeps track of the number of QAStudent applications registered with QAInstructor. The display of that counter is updated whenever the instructor clicks the Refresh button.

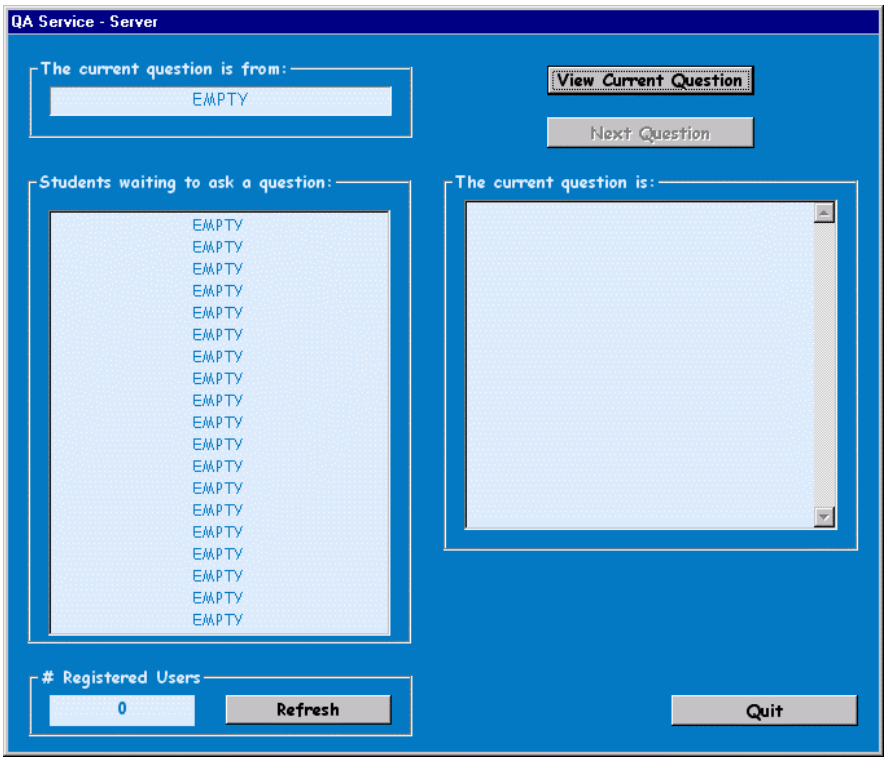

Figure 6 - Initial display of QAInstructor

As students submit questions, their names appear sequentially in the instructor's display. When the instructor clicks the View Current Question button, the question submitted by the student shown in the box labeled The current question is from is displayed in the box labeled The current question is. At this point, the View Current Question button is disabled, the Next Question button is enabled, and the instructor will most likely read and answer the question over the live video stream controlled by the xbindTV applications. When he or she is ready to proceed to the next question, the instructor will click the Next Question button. That action causes The current question is from and The current question is boxes to be cleared. The top name in the Students waiting to ask a question box is then moved to the box labeled The current question is from, the remaining names are shifted one line toward the top of the display, the View Current Question button is enabled, and the Next Question button is disabled. The Quit button allows the instructor to end the QAService session.

QAStudent has a display that shows a list of students waiting to ask a question, and it has controls that allow each student to register with the QAInstructor application, submit a question

to the instructor, delete a question that the student has previously submitted, and quit. These controls generate control messages that are sent to the QAInstructor. Some of these messages cause the state of the display to change, so each registered QAStudent application receives a message from QAInstructor that contains the information needed to update the student's display.

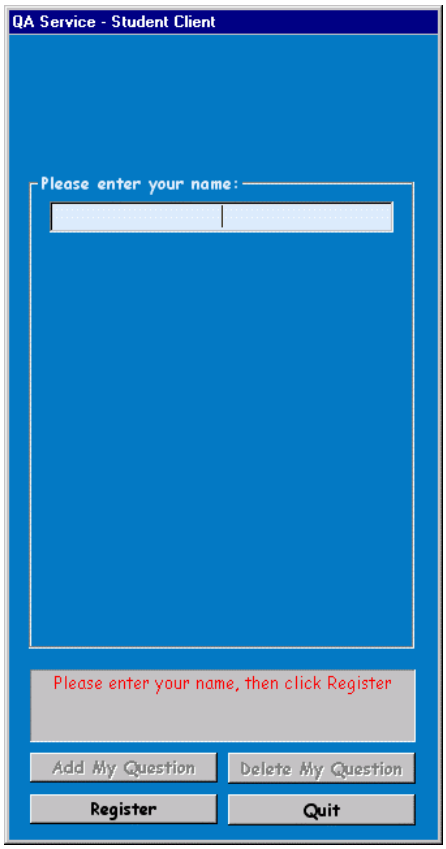

Figure 7 - Initial QAStudent display.

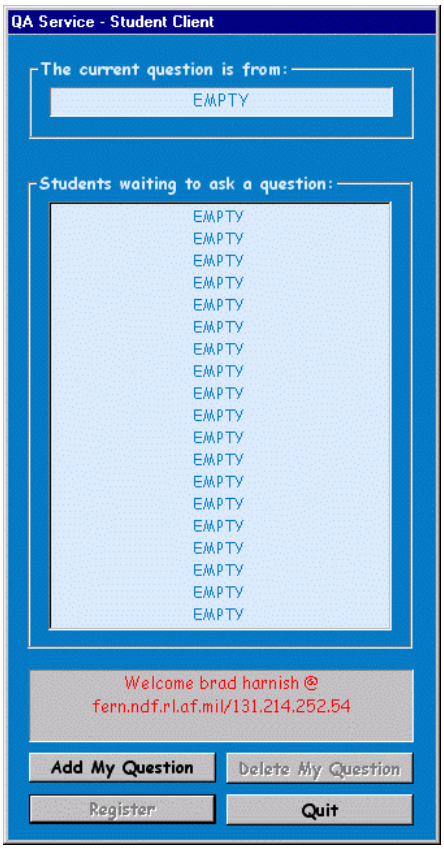

Figure 8 - QAStudent display after registration

The initial state of QAStudent's display is shown in Figure 7. When the student enters their name and clicks the Register button, QAStudent appends the hostname and IP address and sends them all to the QAInstructor application. QAInstructor sends the information that describes the current display content to QAStudent to indicate they have been successfully registered to a QAService session. The newly registered QAStudent display appears as shown in Figure 8.

Once the student has joined the QAService, he or she has the option to ask the instructor a question. A click of the Add My Question button causes a separate window to pop up on the

student's screen in which the student can type in their question and send it to the instructor. QAStudent will generate a local identification number for the question, append it to the student's name and hostname, and send them to QAInstructor along with the question. There, the student's name is added to the Students waiting to ask a question list, the question is saved, and the display information is sent to all registered QAStudent applications to update their displays.

To allow students to ask multiple questions, each submitted question has its own identification number. In the context of each QAStudent application, it is merely an integer. In the larger context of the QAService, each question's identification number consists of the appropriate student's name, their hostname, and the local identification number.

Once the student has submitted a question, the Delete My Question button will be enabled to allow the student to delete it, as long as the instructor has not yet answered it. When the Delete My Question button is clicked, a separate window pops up on the student's screen that displays all submitted questions and their associated local identification numbers. The student is prompted to enter the identification number of the question to be deleted. After doing so and clicking the OK button, the student's name and hostname is appended to the local identification number of the question to be deleted and sent to QAInstructor. There, the name, hostname, and identification number are used to find the appropriate question. If found, the question is deleted, the Students waiting to ask a question list is updated, and the display information is sent to all registered QAStudent applications to update their displays. The student is notified if their question to be deleted can not be found.

The Quit button allows the student to end their participation in the QAService session. If QAInstructor is still running when a student elects to quit, it deregisters the student, deletes all remaining unanswered questions submitted by the student, and sends all remaining registered QAStudent applications updated display information.

The combination of xbindIP, xbindTV, and the QAService provides a basic distance learning application based on common personal computers configured with inexpensive MPEG-1 and MPEG-2 codecs that provide high quality video. Network connectivity needed to support the 6.5Mbps MPEG-2 video streams is expensive and limits its practical applicability to existing

ATM enterprise networks, although the completion of the painfully slow rollout of Asynchronous Digital Subscriber Line (ADSL) services will provide a more ubiquitous platform. Additional web and/or java applications could easily provide the features of a more advanced distance learning package. But even in its present form, the prototype can serve as a useful tool in certain environments. Planned enhancements to the prototype are described in Section 7.

6.0 The Laboratory Testbed

Figure 9 shows the DMIF testbed that has been installed on the Griffiss Business and Technology Park to connect the Programmable Networks Facility of the Air Force Research Laboratory Distributed Information Systems Branch (AFRL/IFGA) in Building 3 and the New York State Technology Enterprise Corporation (NYSTEC) in Building 119. The testbed provides a useful experimental platform for the development of multimedia applications that use the DMIF Application Interface.

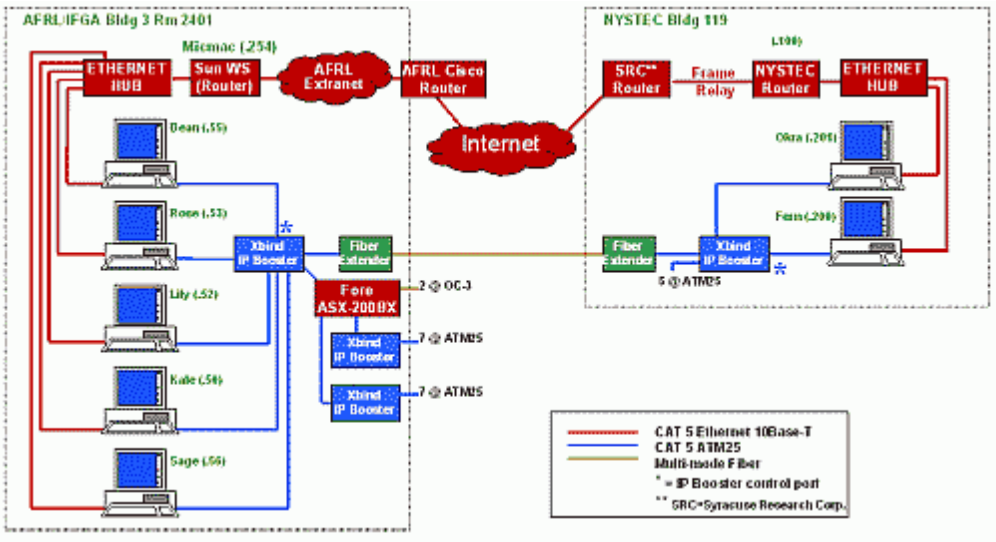

Figure 9 - Griffiss Business & Technology Park DMIF Testbed

As shown, the two sites are connected with two distinct paths - one through the Internet and the other through a 25 Mbps Asynchronous Transfer Mode (ATM25) link. Each site has a cluster of Windows NT™ personal computers that run the xbindIP and xbindTV software, as described in Section 4. Each computer has an ethernet network interface card (NIC) that connects it to the

Internet through an ethernet hub and an Internet Protocol router. In addition, each computer has an ATM25 NIC that connects it to an ATM25 switch. All of these connections are carried over copper (CAT5) cables. The two ATM25 switches are connected by a device at each site that transparently extends the ATM25 protocol over a multimode fiber that connects the two buildings.

Each IPBooster shown is an eight port ATM25 switch that costs about $1,800. They were selected for use in the testbed because they can be controlled by the switch controller component of the xbindIP software which executes in the computer connected to the control port of each switch, shown by asterisks in Figure 9. There are two unused IPBoosters to allow for testbed expansion. A Fore Systems (now Marconi) ASX-200BX switch with four ATM25 ports and two OC-3 ports is also available to provide future connectivity to other sites. But since it can not be controlled by the xbindIP software, it is used to provide static bandwidth (permanent virtual paths and permanent virtual circuits) through ATM networks between IPBooster switches.

The ATM25 NICs are all the same (ForeRunnerLE25, about $150 each) and were chosen because a CCM interface (see Figure 2) was provided by the manufacturer which enabled a virtual switch object in the Virtual Network Device Level to be created to allow the xbindIP software to control them.

All Internet Protocol traffic is sent through the ethernet NICs, and any model that works with Windows NT™ can be used in conjunction with xbindIP and xbindTV. The ethernet NICs are all connected to 10BaseT ethernet hubs at each site. The remaining components of the Internet connection shown in Figure 9 are external to the testbed.

In addition to the video cards, sound cards, and speakers in each computer, they each have an MPEG-2 card installed to enable high quality audio/video content. Sage and Okra are used for source content, so they both have a VisionTech's MVCast MPEG-2 encoder card installed, which each cost about $1,700. These encoders receive analog video from various sources (such as a DVD player, a VCR, or a video camera) depending on the specific demonstration. Bean, Rose, Lily, Kale, and Fern are used as content destinations, so they all have Sigma Design's

RealMagic Netstream2 MPEG-1 and MPEG-2 decoder cards installed, which each cost about $250. The software development kits provided by these codec manufacturers were used to allow the xbindTV software to control them. A limited number of spares are available for testbed expansion.

Besides the xbindIP and xbindTV software, the testbed uses the Microsoft Peer Web Server™ that comes with WindowsNT™, Microsoft Visual J++™ and Visual C++™ integrated development environments, and Microsoft Office™ tools to support collaboration, presentations and documentation.

7.0 Future Directions

As mentioned above, this in-house effort supports the Cooperative Research and Development Agreement (CRADA) with Xbind, Inc. It also supports a CRADA with the New York State Technology Enterprise Corporation, which explains their participation in the Griffiss Business and Technology Park DMIF Testbed. The combined team will continue to investigate the application of networked multimedia services and their management and control technologies against military, government, and commercial requirements. This section of the report summarizes the future direction of the project.

The distance learning application described above has niche utility in the military, government, and commercial markets as it now stands, but enhancements to the application will broaden its customer base and should help attract funding for technology transfer or insertion. The features we will be investigating over the next year are discussed below.

7.1 Voice over IP (VoIP) and the Session Initiation Protocol (SIP)

As mentioned above, the public switched telephone network (PSTN) could easily provide a feedback channel from the students to the instructors. Emerging VoIP technology would allow this type of voice channel over the existing Internet Protocol (IP) networks that already connect the classroom occupants without the need for separate PSTN connections.

We will investigate the augmentation of the existing text-based QAService with voice capabilities based on VoIP technology that supports voice communications services on IP networks using the microphones, sound cards, and speakers included with most off-the-shelf personal computers. Investigations will include various ways to initiate a VoIP sessions in the QAService and will include an investigation of the **Session Initiation Protocol** (SIP) [Draft RFC 2543] and its interoperability with DMIF in the testbed to support other multimedia sessions that can augment the distance learning application.

7.2 Extensible Markup Language (XML)

We will also investigate the use of XML as a means of transferring input parameters and their contextual meaning from xbindTV Station to the program that publishes the course portal on the world wide web server. This will make it easier to extend the basic building blocks of the service to new applications, such as those that support emergency management and response teams, power distribution monitoring systems, so-called smart highway systems that monitor and manage traffic flows, and other generalized command and control systems that could benefit from the integration of digital video distribution and web-based information systems. Java applications or applets could be used to prompt the content provider for customized input parameters, convert it to XML, and send it to the web server where it is converted into the customized web page that provides the user interface to the overall service.

7.3 Wireless extensions

To improve the testbed support for future development of capabilities that enhance command and control applications, we will investigate wireless testbed extensions, such as wireless personal/local area network (PAN/LAN) components and broadband wireless link components.

These wireless extensions could be used to facilitate temporary deployments such as military or emergency management operations. As shown in Figure 10, the broadband wireless components (red) could be used to provide access to other wireless or wired infrastructures remotely located from the deployment site, and the wireless PAN/LAN components (blue) could extend the testbed to mobile users.

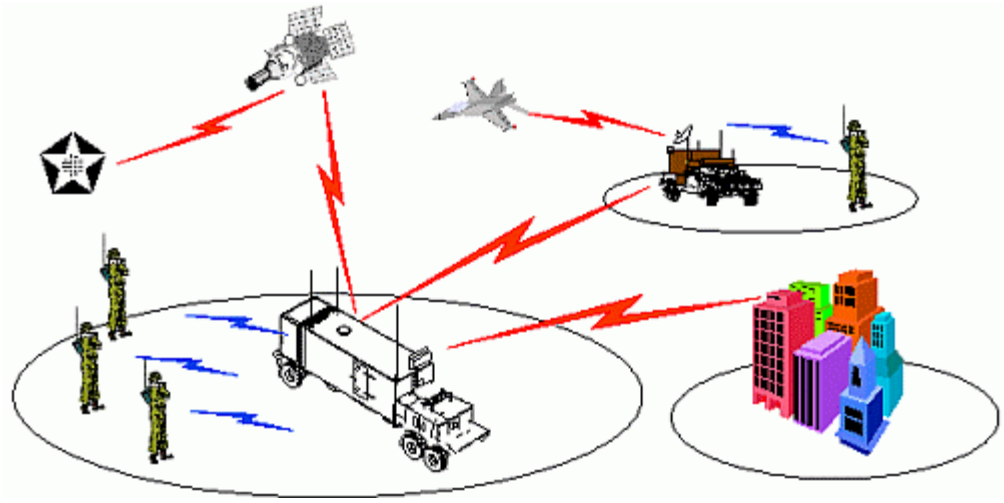

Figure 10 - Wireless network extensions

In NYSTEC's emergency management example, the DMIF platform could be deployed to a disaster area command center to support the field agents and their managers in their assessment and reporting of the damage. Wireless PAN/LAN equipment could be used to provide live video feeds from field agents to the command center. Broadband wireless assets could be used to connect the command center to the local telephone company or competitive local exchange carrier facilities that can provide the necessary wide area network services for coordination with headquarters and other agencies.

Further development of xbindIP to support these wireless interfaces will depend on Xbind, Inc. business interests. Therefore, unless we focus on specific interfaces that are already part of their business plan or funding is acquired to directly support the necessary software development, this investigation will be limited to wireless capabilities that can be integrated into the ATM or IP networks at the link level and operate transparently to xbindIP interfaces.

7.4 Joint Battlespace Infosphere

We will also investigate features that would enable course content and related material to be published in the Joint Battlespace Infosphere (JBI) framework. If JBI users subscribe to this type of information, they could be notified when a course (or any digital video stream and collateral material) becomes available. The investigation will include requirements for metadata that

content providers will need to include in their announcements to allow the subscription to be fulfilled with the simple delivery of the content web page URL.

We will also study the potential utility of DMIF and SIP to support multimedia subscription deliveries in the JBI context. Figure 11 shows a conceptual diagram of network connections that could be supported by the DMIF platform.

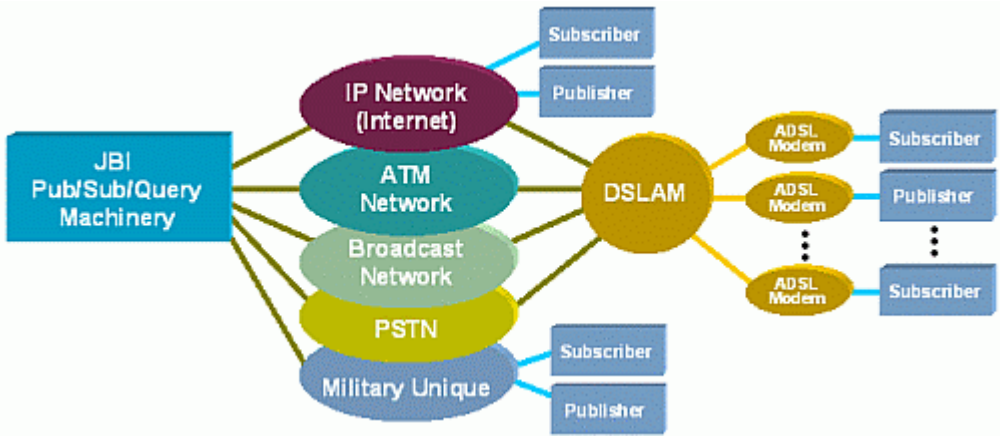

Figure 11 - Network connections in future JBI scenarios

Currently, the JBI fulfills its subscriptions almost exclusively over the Internet, which is suitable for best effort delivery of data-centric products, such as documents, images, data files, and web pages. However, in the future the need for delivering streaming multimedia-based subscriptions will require more stringent delivery bounds (e.g. delay, jitter, reserved bandwidth) on the network infrastructure. A DMIF platform could support military unique systems, the public switch telephone network (PSTN), Broadcast, and native ATM network assets in addition to the Internet while providing a single interface (the DAI) to the JBI system and its users.

This capability could provide the flexibility needed for joint multinational deployments using all available resources without requiring modifications to DAI-based application software, as it is the responsibility of the DMIF system to provide the DAI services over whatever transport medium it supports.

7.5 Rome Research Site enterprise network infrastructure

The DMIF platform is ideally suited for integration into the Rome Research Site (RRS) enterprise network infrastructure using fiber paths between IPBooster switches for high bandwidth traffic with QoS guarantees and standard RRS Intranet or Extranet connections for IP traffic. Because the fiber paths would entail static bandwidth allocations in the form of permanent virtual circuits and or permanent virtual paths, they could be easily isolated from the operational network to minimize risks. If we can find willing participants, we will work with them to extend the testbed to their facility and develop suitable demonstrations. Possible demonstrations include the distribution of live video presentations from the auditorium, for example, to various locations at RRS and the distribution of taped educational or training courses to classrooms or individual desktop personal computers.

Results of these investigations will be included in the Final Technical Report produced at the end of the project.

8.0 Conclusions

The ATM Management and Control APIs project demonstrated the viability of network management and control software based on distributed object technology with a separation between network control and information transport and open interfaces between cooperating and competing entities in this environment. The IEEE Proposed Standard for APIs for Networks and the MPEG-4 Delivery Multimedia Integration Framework (DMIF) and the DMIF Application Interface (DAI) are based on these same concepts.

DMIF and the DAI provide application developers with the flexibility to ensure that their multimedia content is delivered with the quality of service they need while insulating them from the details of the underlying transport technology. This allows their applications to function as designed even if the transport technologies change over time, increasing the life span and flexibility of their applications. The DMIF system is responsible for adapting to the features and/or limitations of new transport technologies to ensure that the DAI services are maintained. At the same time, the abstraction of the streams presented to the DMIF system and the session

protocol accessed though the DAI, also allows the DMIF system to support applications that are not using MPEG-4 coding and synchronization.

The combination of xbindIP, xbindTV, and the QAService provides a basic web-based distance learning application on common personal computers configured with inexpensive MPEG-1 and MPEG-2 codecs to provide high quality video.

The Griffiss Business & Technology Park DMIF Testbed provides a useful experimental platform for the development of multimedia applications that use the DAI. We will be using it to investigate enhancements to our distance learning application based on SIP and XML. We encourage the expansion of the DMIF testbed within the Rome Research Site and will support other development teams that wish to use it.

9.0 List of Acronyms

AAL5 – ATM Adaptation Layer Type 5
ADSL – Asynchronous Digital Subscriber Loop
AFRL/IF – Air Force Research Laboratory/Information Directorate
AFRL/IFGA – Distributed Information Systems Branch of AFRL/IF
API - Application Programming Interface
ATM - Asynchronous Transfer Mode
ATM25 – 25 Mbps ATM standard on copper
BSD – Berkeley Standard Distribution
C-plane – Control Plane
CAT5 – Category 5 copper cable standard
CM – Connection Manager
CPE – Consumer/producer engine
CPFE - Consumer/producer front end
CRADA – Cooperative Research and Development Agreement
DAI – DMIF Application Interface
DMIF – Delivery Multimedia Integration Framework
DSL – Digital Subscriber Loop
DSLAM – DSL Access Multiplexer
IEC - International Engineering Consortium

IEEE – Institute of Electrical and Electronics Engineers

IP – Internet Protocol

ISO – International Organization for Standards

JBI – Joint Battlespace Infosphere

LAN – Local Area Network

Mbps – Mega (million) bits per second

MPEG – Moving Picture Experts Group

MuxP – Multiplexer Port

NetP – Network Provisioning

NIC – Network Interface Card

NYSTEC – New York State Technology Enterprise Corporation

P1520 – IEEE Proposed Standard for Application Programming Interfaces for Networks

PAN – Personal Area Network

PSB – Protocol Stack Builder

PSTN – Public Switch Telephone Network

QAInstructor – Instructor component of QAService

QAService – Question and Answer Service

QAStudent – Student component of QAService

QECI – QoS-enabled Engine Control Interface

QoS – Quality of Service

RRS – Rome Research Site of the Air Force Research Laboratory, Rome, New York

RTP – Real Time Protocol

SIP – Session Initiation Protocol

TCP – Transmission Control Protocol

TP – Transport Protocol

U-Plane – User Plane

UDP – User Datagram Protocol

URL – Universal Resource Locator

VoIP – Voice over Internet Protocol

XML – Extensible Markup Language

XRM – Extended Reference Model