Implementation of Data Storage Media Control and Command (DSM-CC) Core User-to-User Interface for MPEG-2 Bit Stream Transport.

SeongJong Choi*, Yong Han Kim*, JaeWoo Kim*, HoJang Lee*, JaeKyu Shim*, Jae D. Kim**, Jong-Seog Koh**

* Department of Electrical Engineering, University of Seoul (chois@scucc.scu.ac.kr)
** Transmission Technology Research Labs, Korea Telecom

ABSTRACT

This paper describes implementation of the core DSM-CC UU interface. It briefly describes the reference model for the DSM-CC and related standards that should be reviewed for the implementation. The Common Object Request Broker Architecture, Revision 2.0 (CORBA 2.0) is used as a remote procedure call (RPC) scheme for the UU Interface. Entire system was implemented with C++ on Windows NT platforms. The implementation procedure has been decomposed into two tasks. The first task is to implement the Naming Service for service navigation. The Naming Service is one of the CORBA Services that extend the core CORBA specification. A client GUI is implemented for easy navigation among various services, The second task is to construct multimedia server and client for a Video-on-Demand (VoD) system. MPEG-2 Transport Stream is transported via ATM AAL5 using the Windows Socket 2.2 ATM extension API. A GUI enables the user to navigate the service domain and select a program. After the selection the user can control the MPEG-2 stream with VCR-like buttons.

1 INTRODUCTION

One of the main issues that should be resolved for multimedia services is to develop a standard for open application service system. By adopting Common Object Request Broker Architecture (CORBA) as a high-level software bus, Digital Storage Media Command and Control (DSM-CC) specifies open protocol for control and command of bit stream [1]. CORBA is defined by Object Management Group (OMG) and use to implement Remote Procedure Call (RPC). The Digital Audio-Visual Council (DAVIC), whose purpose is to define system architecture for multimedia services, accepted the DSM-CC as a basis for its system architecture [2]. In order to transport MPEG over ATM network, ATM Forum Audiovisual Multimedia Services (AMS) specifies the MPEG-to-ATM mapping [5].

This paper starts with the introduction to the DSM-CC reference model and detailed review of the User-to-User Interfaces. Next, the strategy and research issues for the implementation of the User-to-User Interface are presented. Finally, future plans and possible application domains of the implementations are discussed.

2 BACKGROUND

2.1 DSM-CC Reference Model

The main purpose of the DSM-CC is to provide a set of protocols to manage and control MPEG bit stream. DSM-CC is defined in terms of a functional reference model in Figure 1. The model consists of three entities, Client and Servers (jointly called Users) and Network to communicate each other. A Server pumps bit streams to Clients. Clients are devices such as set top boxes or computers whose main objective is to consume multimedia streams. Servers are entities that provide multimedia contents as well as services such as directory service for navigation.

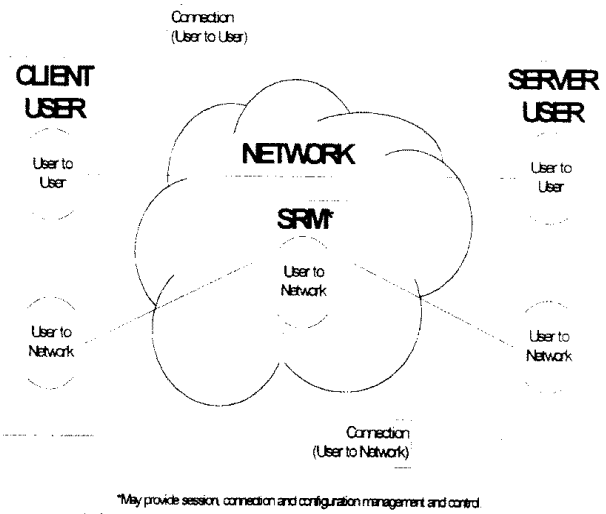

Figure 1. DSM-CC system reference model

2.2 U-U and U-N Interface

A User has two kinds of connections of distinct functionality. One is for User-to-Network (U-N) information and the other is for User-to-User (U-U) information. A U-N information flow is used between Client and Server. U-N information flows between the Network and Client or the Server. The main functions of the U-N connection are configuration, session management and Resource management. The sessions are defined as an associated collection of Resources required to deliver a Service. The configuration feature provide the DSM-CC specific and network specific parameters to the Users (Client and Server)

which are needed for communication with the SRM and the initiation of the first (default) user-to-User Session. The Session and Resource Management features provide management of finite resources needed for U-U session. As shown is Figure 1, the network consists of a Session and Resource Manager (SRM) entity.

2.3 CORBA

CORBA is a distributed object framework proposed by a consortium of over 700 companies called the Object Management Group (OMG). The core of the CORBA is the Object Request Broker (ORB) that acts as the object bus over which objects transparently interact with other objects located locally or remotely.

A CORBA object is represented to the outside work by an interface with a set of operations and attributes. A particular instance of an object is identified by an objet reference. The client of a CORBA object acquires it s object reference and uses it as a handle to make operation calls, as if the object is located in the client's address space. The ORB is responsible for all the mechanisms required to find the objects implementation, prepare it to receive the request, communicate toe request, and carry the reply back to the client. The object implementation interacts with the ORB through either an Object Adapter or through the ORB interface.

The service provided by the server is encapsulated as an object and the interface of an object is described in an IDL. The interfaces defined in an IDL file serve as a contract between a server and its client. Clients interact with a server by invoking methods described in the IDL. Actual object implementation is hidden from the client. Some object-oriented programming features are present at the IDL level, such as data encapsulation, polymorphism and multiple inheritance. CORBA IDL can also specify exceptions.

2.4 User-to-User Interface

The main purpose of the U-U information flow is to achieve generic interactive service request/reply between Server and Client. Requests of the service are performed by a Remote Procedure Call (RPC) protocol carried over a U-U connection. The functions that a U-U connection provides are the Session Gateway Management, Service Gateway Management, Stream Service Control, File Service Control, Database Service Control, and Event Management. These interfaces provide modular building blocks that can be used to for many multimedia applications such as movies on demand, teleshopping, games and distance learning.

Interface definitions for U-U connections are written in the Interface Definition Language (IDL). The IDL provides a grammar for defining the interfaces in terms of behavior of objects. The U-U interfaces written in the IDL are complied by an IDL compiler to produce client and server stubs, a header file used during compilation of the client and server applications, and optional class prototype for coding server implementation.

As shown in Figure 2, there are three distinct interfaces from the viewpoint of U-U Library: an Application Portability Interface (API), a Service Interoperability Interface (SII), and local U-N Interface. (The last one, the local U-N Interface, is not explicitly defined in the DSM-CC. However, during the design and implementation of this project, we found that the explicit use of the interface was very helpful.) The API is defined for client application developers. The SII is defined to allow Clients and Servers from different vendors to operate together. The message in SII is transported to the Network after encoding the Remote Procedure Call (RPC). DSM-CC chose OMG Universal Networked Objects (UNO) RPC scheme and Common Data Representation (CDR) as the default and preferred RPC and data representation schemes, respectively. The UNO and CDR jointly defines the bases of CORBA.

As an example of the local U-N Interface, consider the case when a client wants to start a session. If a Client initiates session attach operation using API, the U-U library determines which operation to execute subsequently. If the client support U-N entity then the U-U library initiates SessionUU interface that is a local U-N interface. Otherwise, it initiates SessionSI SII.

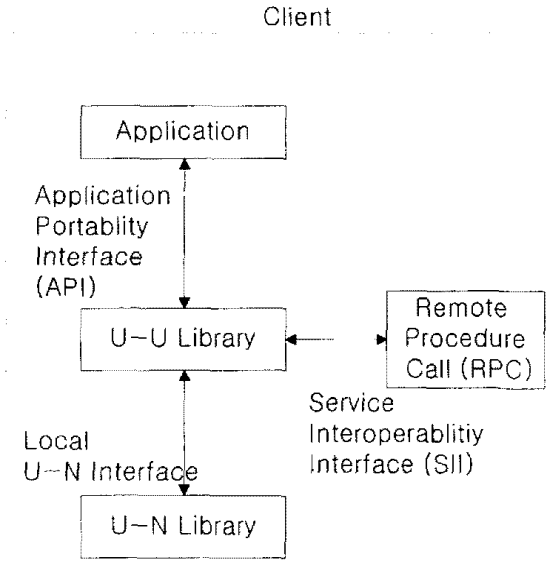

Figure 2. Three types of U-U Interface

DSMCC specifies interfaces either as abstract or instantiable. An abstract interface is only used for other interfaces. A Client cannot access it. Instead, the operations

and attributes of the abstract interfaces are inherited to instantiable interfaces. Therefore the instantiable interfaces are complete, realizable Object Implementation in a Server. The instantiable interface can inherit one or more abstract or instantiable interfaces.

As shown in Figure 3, U-U library sets are categorized in two groups with increasing functionality. Core Interfaces are a foundation for minimal clients. Extended interfaces are defined to facilitate portability and inter-operability. In order to be DSM-CC compliant, Clients and Servers must implement all Core interfaces completely. The following are the brief description of important interface.

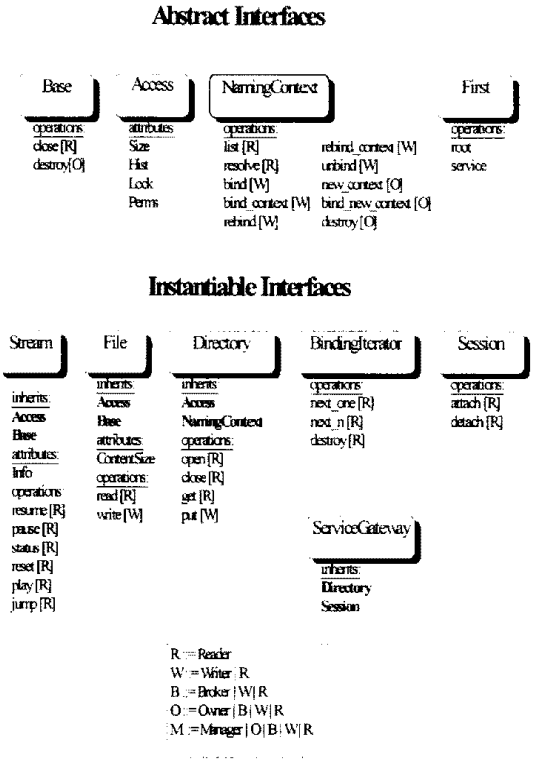

Figure 3. Core U-U Interface

2.4.1 Base

The Base interface provides common operations for deletion of DSM-CC object references and objects. There are two cases when a user no longer needs object reference: when a user wants to delete an object in a server permanently and when a user does not need to user an object. The first case is similar to deleting a file. However, in the second case, all the connection-related resources associated with the object are freed and the object is still alive in a server.

2.4.2 Access

The Access interface provides common description and access control attributes. These include size, version, date, lock and permissions attributes. The AccessRole provides the mechanism for access approval. Three kinds of approval are supported with the Access interface: AccessRole lookup, authentication with password, and encryption of message.

2.4.3 Interfaces for Naming Service -Naming Context, Binding Iterator and Directory

The Directory interface provides a general name space for binding names to Services or data. A ServiceGateway implements the directory interface, and as such provides the primary mechanism for accessing other Services or applications.

Directory defines four kinds of operations:

- Binding a name to an object reference or data value
- Resolving a name to the bound object reference or data value
- Removing a name's binding
- Listing the bindings

The Directory interface inherits the attributes defined by the Access interfaces to allow permission definitions for directories as well as individual Services and data.

For the basic operations involving object references, Directory inherits from the NamingContext interface defined in the CORBA Object Services Naming module (CosNaming). Using the OMG NamingContext interface allows a CORBA environment to readily support DSM-CC. Additional operations defined in Directory is for multiple binding and resolving of object references. This reduces the number of remote procedure call and, eventually, reduces network delay and network traffic.

2.4.4 Stream

Stream primitives are used to emulate VCR-like controls for manipulating MPEG continuous media streams. Streams differ from other datatypes in that, while in play mode, the stream delivery will be governed by an MPEG network flow control mechanism.

Stream pause() and **Stream resume()** behave much like their VCR counterparts. However, each primitive that initiates play mode includes a scale parameter, which controls forward or reverse operation. Position is indicated in Normal Play Time (NPT), which indicates the stream absolute position relative to the beginning of the stream. Application NPT (AppNPT) is used for the application request interface and is specified in seconds and microseconds.

Stream play() enables play from a start NPT position until a stop NPT position is reached. **Stream jump()** provides capability to jump when a stop NPT position is reached to any start NPT position in the stream.

3 IMPLEMENTATION- INTERFACE FOR NAMING SERVICE

In this research, we not only implemented the Core U-U interface but also developed a testbed to verify the various functions of the interface. In this section, we first identify the requirements for implementation.

After the requirement analysis of the implementation, we planned an implementation strategy. The implementation procedure has been decomposed into two tasks. The first task is to implement the Naming Service for service navigation. The Naming Service is one of the CORBA Services, which extend the core CORBA specification. Also, a client GUI is implemented for easy navigation among various services. The second task is to construct multimedia server and client for a Video-on-Demand (VoD) system. MPEG-2 Transport Stream is transported via ATM AAL5 using the Windows Socket 2.2 ATM extension API. A GUI enables the user to navigate the service domain and select a program. After the selection the user can control the MPEG-2 stream with VCR-like buttons

3.1 Developing Tools and Environment

We chose PC with Windows NT 4.0 Workstation as the overall developing and testing platform. The main reason for choosing the PC environment is to utilize the Windows Socket API, which is necessary for MPEG-2 over ATM. The first requirement of the U-U implementation is a developing tool for CORBA environment. We chose ORBIX 2.2 from IONA Technology. The ORBIX provides three distinct functions. The first one is to compile IDL file and create stub files for server and client. The second one is to provide ORB for Remote Procedure Call. The third one is the automatic mapping of the IDL file to C++ prototype for object implementation. Currently, CORBA defines several language mapping of an IDL to an implementation language. Among them are C++, C, Java, Smalltalk etc. We used Microsoft Visual C++ 5.0 as the implementation language for their versatility. In implementing the server object, we built C++ classes with Win32 API. Using Win32 API prohibits source code export to other platforms, but we can take advantages of optimized functions for real-time applications, which are the main concerns of our task.

3.2 Implementation For Naming Service

Directory interface inherits operations of NamingContext interface defined in the CORBA Object Services Naming module (CosNaming). Using the OMG NamingContext interface allows a CORBA environment to readily support DSM-CC. Even though object implementation of the NamingContext is available for free, we built our own NamingContext interface. Since the NamingContext is a basis for the ServiceGateway (L1 Gateway) interface which would eventually provide other important functions such as network management and load balancing.

The essential operation of the naming context is to store and retrieve IOR for user's requests. We used NT file system to store and retrieve IOR with the name defined in the operation parameter. In order to increase the performance of the file operation, we used Win32 API.

4 IMPLEMENTATION - STREAM INTERFACE

The scenario of an actual VoD application session typically consists of the user issuing various commands. At the beginning, a user attaches to a service gateway to find the desired programs. Through Directory interface, the user list, open, or retrieve one of multimedia streams, and then issues various interactive commands, defined in Stream interface, to control the play back of the requested streams. The client-side user interface supports program browsing, and interactive playback. In this section, we present the implementation detail in three parts: Network, Video Server, and Client.

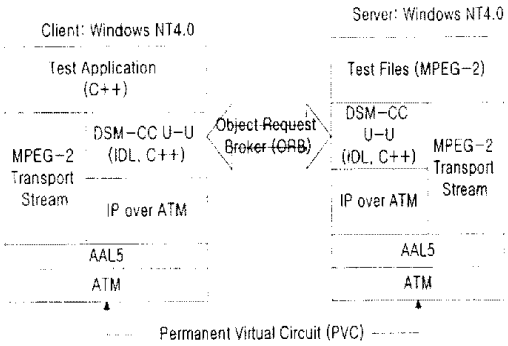

Figure 4. VoD system protocol stack

4.1 Network

The network of our VoD testbed includes ATM cards and point-to-point connection. Fore ATM network interface card, PCA 200, was used for the MPEG over ATM as well as CORBA remote procedure calls. The IP over ATM protocol stack was used for RPC, which delivers control data, such as browsing, retrieval, and file downloads. For MPEG-2 over ATM, Winsock 2.2 API was used to access ATM/AAL5 [3,4]. Using Windows Socket 2.2, we developed C++ classes for client and server, which support operations for mapping constant bit rate MPEG-2 transport packets to/from AAL5-type ATM cells. The Winsock 2.2 has 2 layer protocol architecture in order to enhance multi-protocol stack such as IPX, RSVP and ATM. We have used beta version of Fore ATM Service Provider for the WinSock 2.2 API.

Currently, ATM Forum specifies AAL5 without service specific convergence sublayer (SSCS) transporting constant

packet rate MPEG-2 single program transport stream. The size of CPCS-SDU should be integer multiples of SPTS packet size which is 188 bytes. For example, CPDS-PDU of 384 includes two SPTS packets and 8 byte long trailer. The lower layer, SAR, partitions the CPCS-PDU into eight 48-byte long packets, and finally, 5-byte header is inserted to each packet.

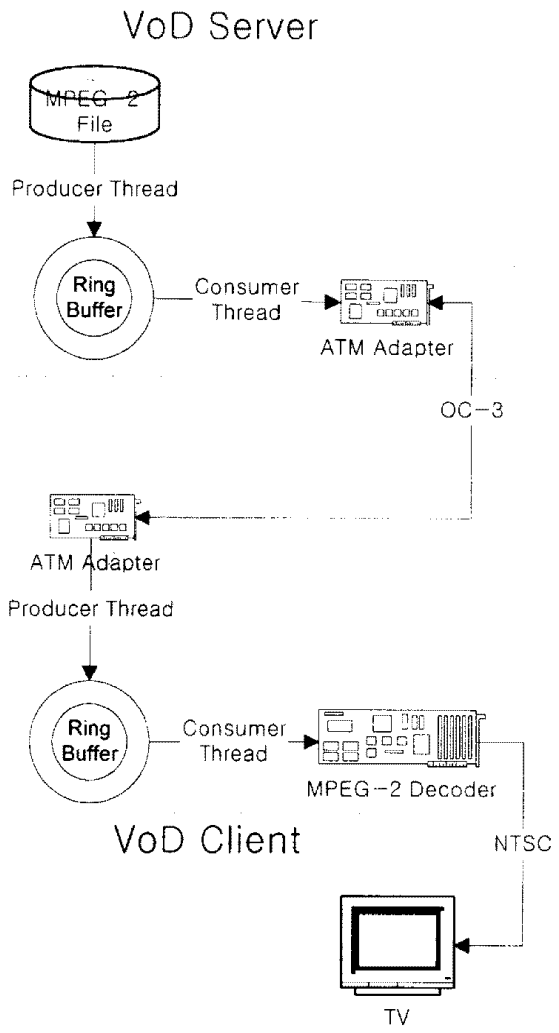

Figure 5. Stream flow in client and server

4.2 Video Server

In this stage of our research, we focused on identifying the design issues to achieve the real-time video server for pumping MPEG-2 TS with constant bit rate. As shown in Figure 5, both server and client have to perform real-time operation to produce and consume constant bit rate transfer of MPEG-2 TS packets. This is the typical producer/consumer multithread scenario. We achieved constant bit rate bit stream using threads both in the server and client. For the synchronization of the threads, we applied semaphore and event in Win32.

After experimenting with the ATM extensions of Windows Socket 2.2 for our Fore ATM adapter card, we found that the number of the TS packets for each Socket "send" call should be larger than a certain amount. The reason is that frequent Socket calls will increase the overhead and hinders the server from the CBR pumping.

4.3 Client

Two applications are developed for Clients. One is a browser for navigating the ServiceGateway. Its Graphic user interface is similar to the Explorer in Windows system, which consists of two panes. All the files are displayed in the left pane with tree structure. By selecting a stream file, a user can view the details of the movie on the right pane such as the playing time, a brief story of the movie, etc. By double-clicking a stream file, a new application starts which displays VCR-like control button.

In order to display the transported MPEG-2 stream in a client, an MPEG-2 decoder is needed. We test on the VideoPlex Decoder. A set of API was also provided with the decoder. The API for control the decoder, and dump transported bit stream to the decoder buffer

5 DISCUSSION AND FUTURE RESEARCH ISSUES

The CORBA tool, ORBIX, was reliable. However, since CORBA is relatively new technology, the lack of learning material, such as tutorial and sample codes, was the major problem to master the CORBA. A strong background in C++ was essential for coding object implementation. Some features, such as "any" datatype, are vendor specific. Also, we found some errors during IDL to C++ mapping, which could be easily fixed. Using many name spaces could cause some confusion during C++ coding. A visual tool for automatic export to visual C++ environment would be helpful for easy development.

We have implemented the NamingContext with file system. It is anticipated that the NamingContext will play important roles in various situations. The NamingContext acts as the source for usage data for a commercial system. Each access to a service might need to be approved and charged by a service provider. Another situation is load balancing, which allows distributing user's service requests to a pool of servers in predefined way. From these reasons, the NamingContext should be reliable and flexible. A database management system may be a good candidate for an implementing tool for the NamingConext.

We have implemented fast-forward, pause, slow, and step for Stream interface. Jumping to an arbitrary position in a MPEG file require an indexing scheme. As stated above, server and client programs are constructed with multithreading. Enhancement of the I/O performance is

another major issue in VoD server implementation. Since Windows NT supports SMP architecture, win32 API that takes advantage of a multiple CPU platform should be take into account. Implementing reverse play is another future research issue.

One of the major issues for multithreading is the synchronization. During the implementation of Stream interface, we found that the state diagram provides a useful design tool for thread synchronization. When developing a multithreading program, a complete state diagram at the initial development stage reduces enormous loads at the coding stage.

As stated above our focus has been to the software architecture for constant bit rate generation and consumption. Also, in case of switched virtual circuit, the ATM switch will cause problems such as delay, and jitters. The main performance factors of the server are throughput and stability, that is, how fast and stable it can pump bit streams to the ATM network. By careful selection of Win32 API and optimal design of software will maximize the both. Optimizing the hard disk access is another issue for better performance. The main concern of clients, like set top boxes, is system resources such as memory and computing power. Constant and jitter-free transmission can reduce the buffer size, will minimize the system resources.

6 CONCLUSION

In this paper, we described the implementation of DSM-CC U-U interface under Windows NT environment for VoD service. As a requirement analysis of the implementation, we identified necessary tools and standards. An implementation procedure was proposed which consists of two steps: implementing interface for naming service, and implementing interface for VoD service. The U-U interface was implemented in C++ language for the following programs: object implementation in a server, client U-U library, and client applications for VoD service. Subsequently, we presented future research issues related to various multimedia services.

The implementation technology used here can be applicable to various fields of multimedia industry such as design of set top box and VoD server, and multimedia communication.

ACKNOWLEDGEMENTS

The work described in this paper was partially supported by Transmission Technology Research Labs, Korea Telecom.

REFERENCES

[1] ISO/IEC 13818-6, "Information technology – Generic coding of moving pictures and associated audio information – Part6: Extension for Digital Storage Media Command and Control", 12-July-1996 (pre-editing release).

[2] DAVIC 1.0 Specification, December 1995.

[3] Windows Sockets 2, Application Programming Interface, An Interface for Transparent Network Programming, Under Microsoft Windows™, Revision 2.2.0, May 10, 1996

[4] Windows Sockets 2 Protocol-Specific Annex Revision 2.0.3, May 10, 1996

[5] Audiovisual Multimedia Services :Video on Demand, Specification 1.1, af-saa-0049.001, March, 1997