INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa • University of Ottawa

An Open Signaling System Over VIVID ATM Switches

By

Li Zhang

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute of Electrical Engineering
School of Information Technology and Engineering (SITE)
Faculty of Engineering
University of Ottawa
Ottawa, Canada

August 8, 1998

Abstract

In recent years, due to the market deregulation and the extensive deployment of broadband communication technologies, such as ATM, the telecommunication society is facing the new challenge of "service competition". The key issue is the flexibility in the creation and deployment of new services in response to the market demand. This in turn poses the need for re-examination of the network software architecture, taking advantage of recent advances in distributed computing, object-oriented analysis and design and other concepts and standards, including CORBA, ODP, etc.

With the endeavor for "open distributed" network software architectures, some new issues, such as "network programmability" and "open signaling", are raised. With the support of Distributed Object Computing (DOC) technology, such as CORBA, it is possible for the communication network to be exploited as a "programmable entity", in which not only the software components but also the hardware resources are to be programmed as "objects" with well defined interfaces. Since the low-level hardware resources are also wrapped with "programmable" software interfaces, the service creation and deployment can be done in a high-level, as well as in a consistent manner, by simply interconnecting (or binding) a set of computational objects.

In terms of easier service creation and deployment, the connection management service can be seen as the basis within broadband networks. Based on the above paradigm, when the interfaces for the control and management of the transport network elements are available, the resource control and management of the transport network can be simply performed by binding the transport network objects. This is so called "open signaling" approach. Since the connection management can also be manipulated as a high-level programmable "object", high-level services creation can be greatly facilitated.

Based on the above observations, this thesis addresses the issues of "network programmability" and "open signaling". Subsequently, the implementation of an "open signaling system" over a real-time environment is presented.

Acknowledgments

I would like to thank my supervisor, Professor N.D. Georganas, for his continuous support and encouragement, and Dr. Ionescu and Dr. Groza, for their extensive discussion and help, during my stay at the Multimedia Communications Research Laboratory (MCRLab) at the University of Ottawa.

I also gratefully acknowledge that the TRIO Industrially Specified Research Program and Newbridge Corp. made this project possible. Special thanks go to Eugene Zywicki, Hossein Sahabi, Chris Anastasiadis and Asif Islam.

I would also like to thank all other MCRLab members, department staff and Newbridge VIVID group members for their kindness and help. Study at the University of Ottawa is truly an unforgettable experience in my life.

Table of Contents

List of Figures

List of Acronyms

ATM	Asynchronous Transfer Mode
AAL	ATM Adaptation Layer
SAAL	Signaling AAL
SVC	Switched Virtual Channel
PVC	Permanent Virtual Channel
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier
OMG	Object Management Group
OMA	Object Management Architecture
DOC	Distributed Object Computing
CORBA	Common Object Request Broker Architecture
COSS	Common Object Service Specification
DII	Dynamic Invocation Interface
SII	Static Invocation Interface
TINA	Telecommunication Information Networking Architecture
TINA-C	TINA-Consortium
ITU-T	Telecommunication Standardization Section of International Telecommunications Union
QoS	Quality of Service
ISDN	Integrated Services Digital Network
B-ISDN	Broadband Integrated Services Digital Network
TDM	Time Division Multiplexing
OAM	Operation, Administration and Maintenance
UNI	User-Networking Interface
NNI	Networking-Networking Interface
IETF	Internet Engineering Task Force
RSVP	Resource reSerVation Protocol
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
RTSP	Real-Time Streaming Protocol
RPC	Remote Procedure Call
DCOM	Distributed Component Object Model
MSS	Multimedia System Services
ODP	Open Distributed Processing
RM-ODP	Reference Model for Open Distributed Processing
DPE	Distributed Processing Environment
BIB	Binding Interface Base
CCSS7	Common Channel Signaling System No.7
CMAP	Connection Management Access Protocol
ONPENSIG	Open Signaling
OPENARCH	Open Architecture
NMTI	Network Management Terminal Interface

Chapter 1

Introduction

1.1 Motivation

The last few decades have seen an astonishing evolution of communication networks. With the advent of high-speed broadband networks, in particular, the deployment of Asynchronous Transfer Mode (ATM) technology, a vast variety of new services has been introduced, such as multimedia services, mobile services and so on. The demand for more advanced and sophisticated telecommunication services is still on the increase. On the other hand, as the result of market deregulation and openness, today's network

operators are experiencing a new era of "service competition" [CHA95a]. As a consequence, rapid and timely new services delivery in response to the market demand becomes a key factor in determining the success of telecommunication service providers [LAZ97]. This implies that the service creation and deployment must be carried out in ever short development cycles and with greater flexibility for change and customization [LIM96]. However, the deployment of new services in the current telecommunications society is proved to be too time consuming, costly and inefficient. Usually, it takes up to several years to deploy new services due to inflexible software systems [DUP94].

To face the new challenges, the flexibility of the network software architecture is recognized to be an essential issue. With such an architecture, creation and deployment of new services can be done in an easy, prompt, and cost-effective manner. Recent advances in distributed systems and transportable software make the re-examination of network software architectures possible [CAM97]. It is well agreed that the future network software architecture should be open and flexible enough to allow for the rapid introduction of new services. It should, by necessity, be constructed as an open distributed system, taking advantage of the advances in the distributed computing and object-oriented analysis and design and other concepts and standards from the telecommunications as well as the computer industry, such as ODP (Open Distributed Processing), CORBA (Common Object Request Broker Architecture), and so on [STEF95].

In recent years, the object-oriented programming paradigm, with its new ideas of data abstraction, inheritance and encapsulation, has been widely adopted in software development. Because this paradigm also provides a good method for modeling complex distributed systems, the adoption of object-oriented approaches in distributing computing is natural and inevitable. This gave rise to the new but very promising area of Distributed

Object Computing (DOC). By providing a high-level infrastructure on which objects may transparently make requests and get responses over communication networks, DOC technology not only provides an effective solution for the software integration over a heterogeneous environment [KIN96] but also simplifies the development of distributed software by alleviating the inherent heterogeneity and complexity of the underlying networks [SCH97a]. OMG's (Object Management Group's) CORBA (Common Object Request Broker Architecture), as an industry standard, is being more and more adopted in the development of open distributed systems.

The adoption of Distributed Object Computing technology in the communications community leads to the new issue of "network programmability", in which the communication network is exploited as a programmable entity [CHAN97]. As we know, communication networks can be considered as being composed of a collection of hardware and software entities. With the support of Distributed Object Computing technology, such as CORBA, it is possible for not only the software components but also the hardware resources to be modeled (or programmed) as computational "objects" with well defined interfaces, which provide the capabilities for controlling, accessing or managing the hardware or software entities. The interaction among the objects is supported by CORBA infrastructure. Therefore, since the low-level hardware resources are also wrapped with "programmable" software interfaces, the service creation and deployment can be done in a high-level programming paradigm, as well as in a consistent manner, by simply interconnecting (or binding) a set of computational objects. In this way, the introduction and deployment of new services can be greatly facilitated and software reuse can be largely achieved.

Based on these, a generic "open" service model reflecting the operating structure of the future communication services industry is proposed in [LAZ97] by A. Lazar (see Figure 1.1).

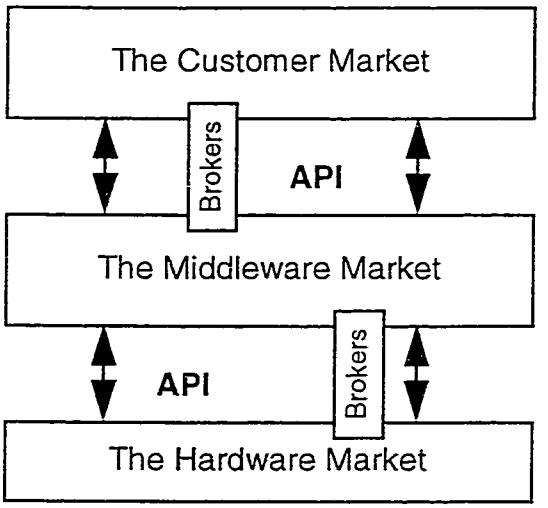


Figure 1.1 The Layered Market Model

This model consists of three layers with each layer representing a market. At the bottom layer, is the hardware market including equipment manufacturers and vendors who provide hardware for building the basic communication infrastructure. Since the hardware entities are all modeled as computational and programmable objects, the APIs that the equipment manufacturers and vendors provide to the upper layer consist of the open interfaces being used to control and access the hardware entities. This is so called "open network control".

In the second layer lies the middleware service provider to provide network services. Based on the APIs, i.e. open interfaces for controlling and managing the hardware resources, provided by the hardware market, different management and control policies of the network can be applied. This enables customers to configure their systems in a manner best suited to their needs, so that more efficient utilization of network resources can be achieved. The APIs provided in this market are sufficiently high to allow for the development of any customer level services.

On the top layer lies the customer service provider. Based on the network services provided by the middleware layer, the customer service provider can freely compete to

bundle, integrate and develop services to meet the customers demand, as we can see the flexibility in the service creation and deployment within such a service model.

Until recently, a conceptual "open" software architecture, called TINA (Telecommunication Information Networking Architecture), with the aim of rapid delivery of telecommunication services in the broadband, multimedia and mobile era to meet the increasing market demand, is being addressed by TINA-C (TINA-Consortium). In addition to the research endeavor by TINA-C, as a proof of concept of the new network software architecture, the COMET group at Columbia University gave an implementation of an open programmable broadband kernel, called Xbind, with the focus on the multimedia communication networks. The TINA and Xbind architecture will be described in detail in Chapter 3.

Along with the exploration of network programmability in future communication networks and their software architectures, "open signalling" is another issue raised. As mentioned above, the DOC technology makes it possible for the hardware entities in the communication networks to be wrapped as high-level programmable "objects", and capabilities for the control and management of them are given by the "object" interfaces. As such, the resource control and management of the transport networks can be simply performed by interconnecting (or binding) a set of objects which model the transport network elements, such as switches, hubs and routers. This implies a new "signalling" (or connection management) approach for communication networks, that we called "open signalling".

Networking signalling has been playing an important role in communication networks in terms of efficient use of network resources and service offerings [WU95] [MIN89]. Especially in ATM based broadband communication networks, the connection management service can be considered as the basis for the creation of other advanced

services. For easier service creation and deployment in the future network software architecture, the connection management service needs to be designed for flexibility and scalability [CHAN97]. From this point of view, the current signalling stack-based protocols, such as Q.2931, which is being used as the ATM signalling standard, are proved to be rigid and complex. In contrast, the "open signalling" approach is much more flexible and scalable for service creation and deployment.

In reality, the concept of "open signaling" was proposed by the COMET group at Columbia University with the launch of Xbind project in 1994. Being targeted at a "broadband kernel" for multimedia communication networks, the emphasis of Xbind "open signaling" is put on the control of ATM switching nodes. Along with the development of Xbind project, the COMET group first gave an implementation of an "open signalling" system using FORE Systems switching technology around 1995. The early products of ATM switches were UNIX OS-based. In recent years, with the high-performance real-time operating system available, such as VxWorks, pSOS, most of the ATM switch vendors have been gradually migrating their products to real-time based OS. Due to the advantage of real-time OS scheduling features, it is envisaged that in the future most of the ATM switches in the market, although from different vendors, will be real-time OS based. In this case, deployment of such an "open signalling" system over real-time environment is of much more significance.

Based on the above observations, as well as the Xbind system developed by the COMET group, the thesis work gives an implementation of an open signalling system over real-time OS based VIVID ATM switches.

1.2 Thesis Outline

Since our work is also greatly motivated by research on multimedia communications, Chapter 2 of this thesis will review some issues concerning Distributed Multimedia

Systems. The focus is on how distributed multimedia computing is supported by different aspects of technologies. These include: underlying networking technologies, such as ATM; Internet Protocols, such as ST-II, RTP/RTCP and so on; Distributed Object Computing technology, such as CORBA; and CORBA-based middleware issues.

Chapter 3 will detail the "network programmability" issue of future communication networks. First, the concept and structure of the information networks will be introduced. Followed by that, two "open" software architectures, i.e. TINA and Xbind, for the easier service creation and deployment, will be presented.

Chapter 4 presents an overview of the signalling issues in communication networks. In this chapter, first, we will see how the signalling system is evolved over different generations of communication networks, from the early telephony networks to the current broadband networks. Then, the "open signalling" approach deployed for the future broadband networks will be discussed.

Chapter 5 gives a detailed implementation of an open signalling system over VIVID ATM switches. Chapter 6 sets out the conclusions and offers suggestions for the future work.

1.3 Main Contributions

The main contribution of this research is the implementation of an "open signalling system" over VxWorks real-time OS based VIVID ATM switches using CORBA. Our work is based on the Xbind system, but because of the uniqueness of the real-time OS, some new challenges are faced. Our work for the implementation of such a system can be divided into the following aspects and stages:

- Deployment of CORBA/Orbix over a VxWorks real-time OS environment.

- Development of open APIs for VIVID ATM switch control and management. The switch fabric control APIs provide interfaces for reading, adding and removing entries from the ATM switching tables. These APIs provide the building blocks to be used by higher level unicast and multicast connection management algorithms. The management APIs, to a minimum degree, provide interfaces for managing the switch node.
- Development of a NodeServer which resides on the VIVID ATM switch and interacts with the VxWorks real-time operating system of the switch.

Chapter 2

Multimedia Communication Systems

2.1 Introduction

"Multimedia" can be regarded as one of the "hottest buzzwords of the time" [MIN93]. The advent of "multimedia" has greatly changed the way of computer-to-human and human-to-human communication. For example, the mode of the computer-user interaction has transcended the limitations of monotonous style, such as silent text and graphics, to the lively audio and video mode.

Over the past decades there has been enormous progress in the area of multimedia systems and their applications. Early efforts mostly centered on the stand-alone multimedia workstation and associated software systems and tools, such as music composition, computer-aided learning and interaction video [FUR95]. In recent years, with the deployment of high-speed networks and powerful end-systems, multimedia systems have gradually migrated from stand-alone systems to more complicated multimedia communication systems or so called, distributed multimedia systems.

In general, the advent of the distributed multimedia systems is due to the dramatic concurrent advances in three technological fields: computing technology, compression techniques and networking technology [FLU95].

The progress in computing technology lies in the great increase of processing power and storage capacity of the modern computer systems. This is essential for multimedia systems to handle audio and video, i.e. to digitize, compress and decompress a large amount of audio and video data within stringent time constraints and store them in memory while being processed.

Compression techniques also play a key role in multimedia computing. To some degree, there would be no multimedia today without the progress in compression algorithms [FUR95], such as MPEG and JPEG, and their implementations. The use of compression techniques can effectively reduce the storage volumes of audio, video and image, as well as limit the bit rate necessary to transmit them over communication networks.

In addition to the increase of the computer power and the use of compression techniques, the advances in the networking technologies is another important as well as crucial factor [FUR95] [FLU95]. Even if the data are compressed, networked multimedia systems require high bandwidth, low latency and low jitter. This implies that the

transmission of multimedia information places heavy requirements on the underlying networks. The recent worldwide development and deployment of broadband networks, especially the ATM technology, have greatly stimulated the development of distributed multimedia systems.

Specifically, as defined in [STE95a], distributed multimedia systems are characterized by the integrated computer controlled generation, manipulation, presentation, storage and communication of independent discrete and continuous media. Usually such systems consist of two or more computer nodes equipped with multimedia devices interconnected via communication networks. The handling of isochronous media such as audio and video makes them distinguished from the traditional information processing systems [DAA97]. With the deployment of distributed multimedia systems, a new variety of networked multimedia applications has been created, such as desktop video-conferencing and video on demand. As a result, distributed multimedia computing is receiving considerable attention and gaining increasing popularity.

Currently, networked multimedia applications can be generally classified into the following three categories [STE95a] [DAA97]:

- Conversational multimedia applications include multimedia conferencing systems and computer supported collaborative CSCW environments. Conversational multimedia is carried out in a live mode and requires time constraints.

- Presentational multimedia applications are retrieval based applications such as News-On-Demand or Video-On-demand. Presentational multimedia requires distribution servers which supply prerecorded stored multimedia documents to the users' workstations.

- Multimedia mailing applications are store and forward applications with no specific time constraints during the delivery of multimedia data. Multimedia Fax and Multimedia Mail are main examples.

2.2 Supports for Distributed Multimedia Computing

With no doubt, today's distributed multimedia applications are gaining more and more popularity with increasing demand. By handling isochronous media such as audio and video, distributed multimedia applications distinguish themselves from conventional text and graphic applications in the following ways. First, multimedia applications generate a larger amount of data than that of conventional applications. Second, multimedia applications have real-time and error rate constraints so that data must be delivered within a tolerable time period. Third, multimedia applications usually have Quality of Service (QoS) requirements.

In recognition of these, to better support the deployment of distributed multimedia systems, some issues have been addressed from the following aspects.

2.2.1 Network and End-Systems

Because of the voluminous and continuous nature of audio and video data [HAF97], distributed multimedia applications place much more demands on the underlying networks and end-systems.

On the end-systems side, great CPU processing power is required for timely compression and decompression of multimedia data. The modern computer architecture must provide high bus bandwidth and fast and efficient I/O [FUR95]. Additionally, to meet the real-time constraints of multimedia data, the conventional operating systems such as UNIX, which were designed to offer fair use of system resources among

competing processes, become inadequate. There is a great demand for high-performance operating systems, which can support new data types, real-time scheduling, and fast-interrupt processing [STE95b]. Modern micro-kernel based real-time operating systems, such as Chorus [HER88], Mach [ACC86], VxWorks [WIN93] and pSOS have been developed and many real-time enhancements to the existing UNIX based systems have also been made. For instance, the kernel of SunOS has been modified to be fully pre-emptive.

On the network side, advanced networking technologies are needed to provide high bandwidth, low latency and low jitter required for timely and continuous delivery of multimedia data. Generally, multimedia applications generate orders of magnitude more data than conventional applications. Inevitably they require a very high bandwidth. Besides, continuous media have an implied rate of play-back and Quality of Service requirements that have to be followed, otherwise the integrity of application is destroyed, i.e. the information that was intended to be presented can be lost or misunderstood [SIQ97]. However, most legacy networks, which are mainly for low-bandwidth and non-critical applications, provide neither bandwidth nor timing-constraints required by distributed multimedia applications. Consequently, the support for continuous multimedia has become one of the main driving forces behind the endeavor for more advanced communications technologies, such as ATM. The communication networks have experienced a revolution from Ethernet, Token Ring, FDDI, fast Ethernet to today's ATM and Gigabit Ethernet.

The conventional shared-medium LANs, such as Ethernet and Token Ring, are not suitable for multimedia communications [GEO96] [ACA94]. Ethernet networks are designed to process data packets and operate in contention modes, which only provide 10Mbps and their latency and jitter are unpredictable. Token Ring networks provide 16Mbps but the predictable worst-case latency can be very high.

To better support multimedia applications, much effort has been made to enhance the performance of conventional Ethernet and Token Ring. As a result, some new technologies were invented, which included [FLU95]:

- FDDI. It is also shared-medium based but an order of magnitude faster, i.e. 100Mbps, than conventional Ethernet and Token Ring.
- Fast Ethernet. Keeping the Ethernet philosophy but improving its speed to 100Mbps.
- Isochronous Ethernet. Adding on top of Ethernet extra bandwidth that has the property of supporting isochronous communications.
- FDDI-II. Adding isochronous communications on FDDI rings.
- ATM LAN. A new approach, different from the shared-medium technology.
- Gigabit Ethernet. A new Ethernet standard of 1Gbps.

In recent years, the fast switching and Quality of Services (QoS) support of ATM technology make it much more attractive to multimedia applications. As a result, more and more multimedia applications are being conducted based on ATM principles. Here we give a description of ATM technology.

2.2.1.1 ATM technology

ATM (Asynchronous Transfer mode) is a communications standard adopted by the ITU-T (Telecommunication Standardization Section of the International Telecommunications Union) as the core transport mode for the Broadband Integrated Services Digital Network (B-ISDN) [HAN94].

In the middle 1980s, the telecommunications world started the design of a networked technology that could act as a unifier to support all digital services, including

low-speed telephony and very high-speed data communication networks [FLU95]. This is the so called broadband Integrated Services Digital Network (B-ISDN). Consequently, the Asynchronous Transfer mode (ATM) was selected as the transport mode for it.

The purpose of ATM is to provide a high-speed, low-delay multiplexing and switching network to support any type of user traffic, such as voice, data, and video, through the use of fixed-size packets, called cells [BLA95]. Therefore, ATM is known as a cell relay technology. Each ATM cell, consisting of 53 bytes of data, of which 5 bytes are used by the ATM header, is identified with VPIs (Virtual Path Identifiers) and VCIs (Virtual Channel Identifiers). An ATM network uses these identifiers to relay the traffic through high-speed switches. ATM is "asynchronous" in the sense that cells containing information from an individual user do not have to appear in the transfer stream at predicable times, as they would be in traditional Time Division Multiplexing (TDM) [MIN93]. This allows for dynamic bandwidth allocation on demand in ATM networks, such that the network delay over ATM networks is smaller than that on a traditional packet network.

In addition to the ability to provide high-speed transporting and switching of multiple user streams, which may range from real-time data such as voice and high-resolution video, to non-real-time traffic, what makes ATM most ideally suitable for distributed networked multimedia applications is that it can provide different classes of Quality of Service (QoS) to users. Specifically, because of the connection-oriented nature of ATM, during the connection establishment process, the user can specify the desired QoS based parameters such as maximum number of cells per second, maximum end-to-end delay, and maximum jitter allowed. A connection is refused when the network can not support the required QoS level.

To well understand ATM networks, let's have a look at the Reference Model of B-ISDN, which is shown in Figure 2.1 [BLA95] [HAN94] [KAW91].

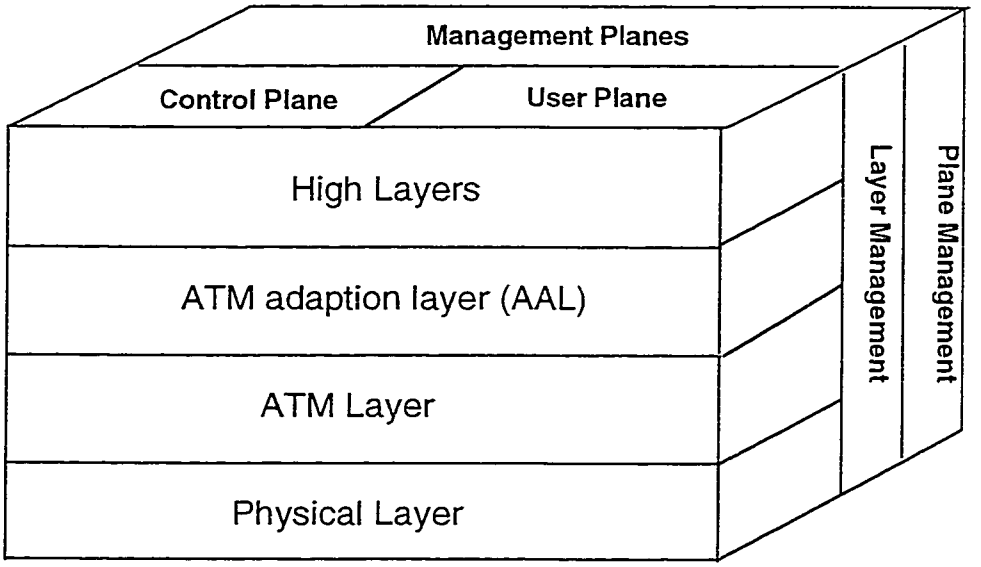


Figure 2.1. ATM and B-ISDN reference model

The whole model consists of three planes, which are the User Plane, Control Plane and Management Plane. The User Plane is responsible for transporting user information and flow control. The Control Plane is responsible for signaling, i.e. dealing with setting up a network connection, managing the connection and releasing the connection. The Management Plane has two functions: Layer Management and Plane Management. The Plane Management is used for coordination of all the planes, while the Layer Management is needed for managing the entities in the layers and performing operation, administration, and maintenance services (OAM).

Pragmatically, the model can also be viewed as different layers (see Figure 2.2):

At the lowest level is the Physical Layer, which performs bit-level functions, adapting the cell-based approach to the transmission medium.

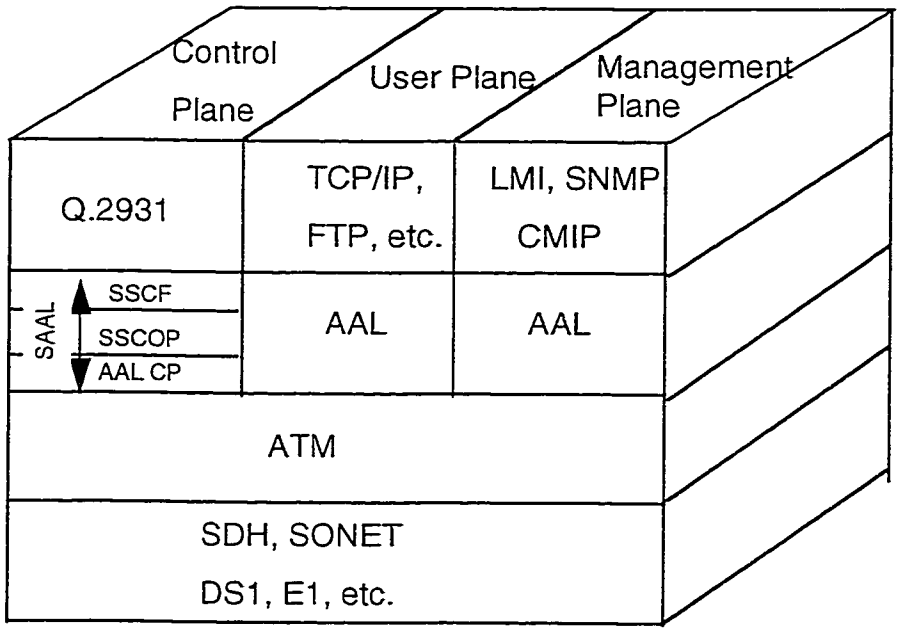


Figure 2.2 B-ISDN layers

The ATM layer is the layer above the physical layer, mainly responsible for the management of the sending and receiving of cells between the user node and the network node. It adds and processes the 5-octet cell header.

The ATM adaptation layer (AAL) is designed to support different types of applications, and different types of traffic, such as voice, video and data. The AAL layer plays a key role in the ability of an ATM network to support multiple applications. The AAL layer is subdivided into the segmentation and assembly (SAR) sublayer and the Convergence Sublayer (CS). The essential functions of the SAR sublayer are, at the transmitting side, segmentation of higher layer PDUs into a suitable size for the information field of the ATM cell and at the receiving side, reassembly of the particular information fields into higher layer PDUs. The CS is service dependent and tailored to support different types of applications.

In addition, ITU-T proposed a service classification which is specific to the AAL layer. This classification was made with respect to three parameters: timing relation, bit

rate and connection mode [HAN94]. Because not all possible combinations make sense, only four traffic classes of service (namely A, B, C and D) were originally identified by the ITU-T, each one providing a data flow appropriate to a different set of applications. For each class of service a corresponding AAL type (AAL1 for class A , AAL2 for class B, and so on) is defined. See Table 2-1.

Class of Service	A	B	C	D
AAL	AAL1	AAL2	AAL3	AAL4
Timing Relation	Preserved		Not Preserved	
Bit Rate	Constant	Variable		
Conn. Mode	Connection Oriented			Connectionless
Applications	Circuit emulation	Compressed audio/video	Virtual circuit services	Datagram service LAN emulation

Table 2-1. Original AAL class of services

As an example of a Class A service is circuit emulation, which is intended for Constant Bit Rate (CBR) voice and video applications. An example of Class B is Variable Bit Rate (VBR) video traffic used in a teleconference. Class C and D are targeted for connection-oriented or connectionless data transfer applications, which usually have no particular timing relationship requirements [GEO96].

Later on, some changes were made. AAL 3 and 4 were merged because of their similarities, giving rise to AAL3/4. AAL5 was created to provide a more efficient interface for class C services. AAL5 is also used by a new class, known as Available Bit-Rate (ABR), which is suitable for delay-tolerant applications. In addition, a new class, called Unspecified Bit Rate (UBR), is introduced for the provision of cell-relay service for applications with minimum QoS requirements, i.e. delay-tolerant and subject to cell loss [SIQ97]. The new AAL layer and its services can be described as in Table 2-2.

Class	X	A	B	C	D
AAL	AAL0	AAL1	AAL2	AAL5	AAL3/4
Time	Not Preserved	Preserved		Not Preserved	
Bit Rate	Constant	Variable			
Con Mode	Connection Oriented				Conn-less
Applications	Unassigned Bit Rate	Circuit emulation	Compressed audio/video	Available Bit Rate	Conn-less service

Table 2-2. New AAL class of services

The higher layers of the B-ISDN model mainly contain some high-level protocols. For instance, in the Control Plane, together with the Signaling AAL (SAAL), the Q.2931 signaling protocol is defined, which is used to set up connections in the ATM network. SAAL supports the transport of the Q.2931 messages between any two machines to set up SVCs (Switched Virtual Circuits). Moreover, user and applications-specific protocols, such as TCP/IP or FTP, and network management protocols, such as SNMP and CMIP, are defined in the User Plane and the Management Plane, respectively.

Compared with the OSI Reference Model, there is not an exact mapping between them. But it can be seen that the ATM model goes beyond the OSI model by introducing the concept of planes subdivided into layers in a three dimensional structure [SIQ97]. However, because ATM is a fundamental technology for B-ISDN, which aims at the unification of low-speed telephony as well as high-speed data communication networks, the ATM network is still based on the concept of low intelligent CPE (Customer Premise Equipment), UNI (User Network Interface) and NNI (Network Node Interface), which are originally adopted in the telephony networks. A simple and possible topology of ATM networks can be described as follows in Figure 2.3.

In short, although ATM technology is not been completely standardized, it is being widely adopted and evolving very rapidly. And also because it can be applied to a range of services, including real-time services such as voice and video, it is becoming the best candidate for distributed multimedia computing.

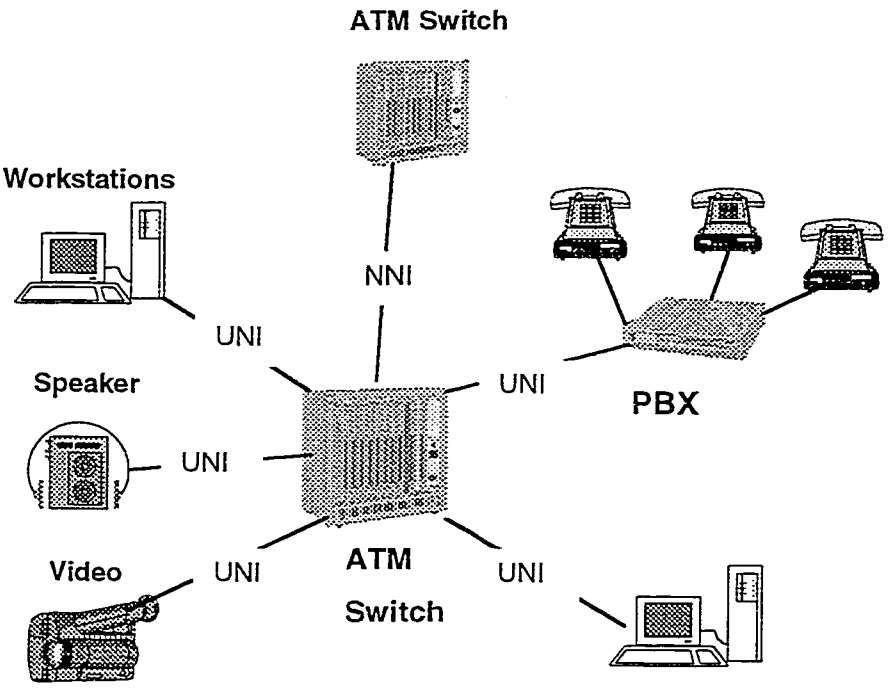


Figure 2.3. A possible ATM network topology

2.2.2 Internet Protocols

In parallel with the endeavor for networking technology and new real-time operating systems to support distributed multimedia computing, the provision of new types of protocols in the Internet community was another great effort.

With the worldwide deployment of Internet, running multimedia applications over it is very attractive. However, the best-effort nature of it would be the major drawback of Internet in the support of real-time transmission of multimedia streams such as audio and video [FLU95]. Therefore, to better support multimedia applications over Internet, some new protocols are being introduced by the IETF (Internet Engineering Task Force), while retaining the existing best effort service.

The new enhanced suite of Internet protocols can be described in Figure 2.4 [SIQ97].

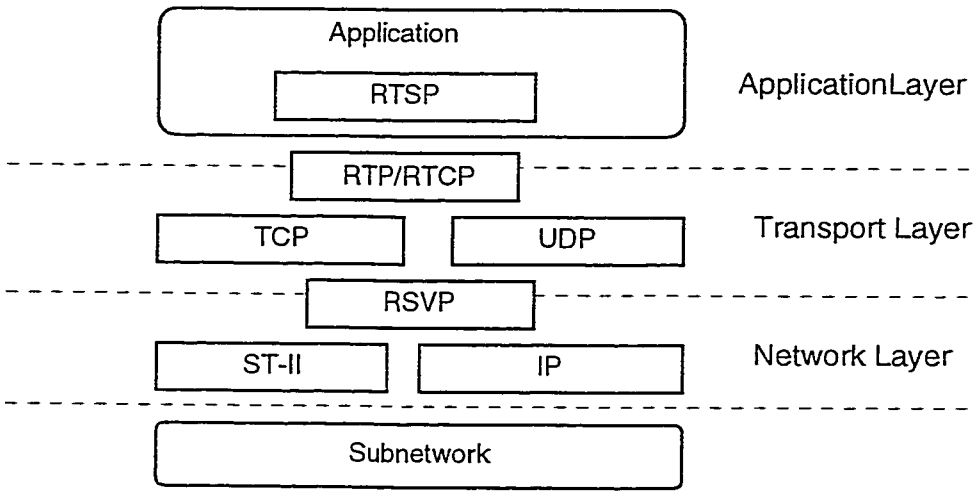


Figure 2.4. New Suite of Internet Protocols

• ST-II

The ST-II protocol [FLU95] [SIQ97] is a connection-oriented network protocol designed to coexist with IP. ST-II has been designed to support continuous media streams. A typical distributed multimedia application would use both protocols: IP for the transfer of traditional data and control information, and ST-II for the transfer of digital audio and video [GEO96]. The ST-II protocol provides a call set-up facility, with which the hosts can declare connections to the network before exchanging data. A set of Quality of Service (QoS) parameters such as minimum required bandwidth, mean end-to-end delay, and delay variation can be specified. When a call originating from a host is received, the ST-II network establishes a virtual link, and resources such as bandwidth and buffers along the virtual link are reserved or allocated. Once the link is set up, the later transmission of continuous media follows the virtual link, while the transmission of other control messages follows other channels without consuming the reserved resources. An associated control message protocol, called Stream Control Message Protocol (SCMP), is used for initiating and modifying connections. ST-II has demonstrated good performance in the support of real-time packet audio and video streams. In addition, the ST-II protocol also supports multicast.

• **RSVP (Resource reSerVation Protocol)**

The RSVP protocol [ZHA93] is designed to improve the support of real-time applications by IP networks. It allows applications to request specific quality of service (QoS) based on the establishment of data path(s) over IP, in which the reservation of resources on each network node can be made when necessary. The selection of the data path depends on the underlying routing protocol to determine where RSVP should carry reservation requests. Uniquely, the resource reservation of RSVP is receiver-oriented in that the receiver of the data is responsible for the initiation of the resource reservation.

The "receiver-oriented" design principle enables the RSVP protocol to accommodate heterogeneous receivers in a multicast group and allows them to make reservations specifically tailored to their own needs. This leads to the primary scalability nature of RSVP. As a result, the multipoint service provided by RSVP can be scaled to very large multicast groups. RSVP is more suitable for applications of audio and video broadcasting.

• **RTP/RTCP (Real-time Transport/Control Protocol)**

RTP is intended to provide end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over unicast or multicast network services [SCH96]. Some high-level services provided by RTP include framing, multiplexing, demultiplexing, encoding, synchronization, error detection and encryption [SIQ97].

Usually, user applications run RTP on top of UDP/IP to make use of its multiplexing and checksum services. But efforts have been made to make RTP transport-independent so that it could be used on other protocols such as TCP. RTP itself does not address the issue of resource reservation or quality of service control;

instead, it relies on resource reservation protocols, such as RSVP. Additionally, a control protocol, called Real-time Transport Control Protocol (RTCP) is used to augment its functionality by allowing monitoring of the data delivery to a large multicast networks and providing feedback on the quality of data distribution.

RTCP is a control protocol that works in conjunction with RTP. It provides support for real-time conferencing for large groups within the Internet, such as source identification and multicast-to-unicast translators. RTP and RTCP are standardized in RFC 1889 and RFC 1890.

• RTSP (Real-Time Streaming Protocol)

The application-level Real Time Streaming Protocol (RTSP) [SCH96], aims at providing an extensible framework to enable controlled delivery of real-time data. Sources of data can include both live data feeds, such live audio and video, and stored content, such as pre-recorded events. It is designed to work with established protocols such as RTP, HTTP, and others to provide a complete solution for streaming media over the Internet. It supports multicast as well as unicast. It also supports interoperability between clients and servers from different vendors.

RTSP has been submitted for consideration as an Internet standard to the Internet Engineering Task Force as a proposed standard protocol for Internet multimedia streaming in one-to-many applications. Through this process, RTSP will allow interoperability between client-server multimedia products from multiple vendors. Allowing client and server software from multiple vendors to interoperate will give users more flexibility and choice.

RTSP is considered more of a framework than a protocol. It is intended to control multiple data delivery sessions and provide a means for choosing delivery

channels such as UDP, TCP, IP multicast and delivery mechanism based on RTP. Control mechanisms such as session establishment and licensing issues are also being addressed. RTSP is being designed to work on top of RTP to both control and deliver real-time content. RTSP can be used with RSVP to set up and manage reserved-bandwidth streaming sessions.

Apart from the protocols that mentioned above, to support real-time multimedia applications is also been addressed in the next generation of IP version 6.

2.2.3 Distributed Object Computing

As mentioned above, advances in several technologies are making distributed multimedia systems technically and economically feasible as witnessed by the proliferation of distributed interactive multimedia services like video-on-demand and video conferencing. In essence, due to the heterogeneous and distributed nature of the distributed multimedia computing, distributed computing technology surely plays an important role in the development of distributed multimedia applications.

Due to the versatility and complexity of the underlying networks and computing platform, dealing with the heterogeneity of the distributed systems is rarely easy. In particular, the development of software applications that support and make efficient use of heterogeneous networked systems is very challenging [VIN97]. For so many years, the low-level BSD Socket and Remote Procedure Call (RPC) mechanism can be seen as the dominated technology for distributed computing [COM91]. Usually, such systems lack flexibility, extensibility, reusability, and interoperability, so that services creation and later modification is time consuming, costly and inefficient [DUP94]. Thus it is hard to meet with them the increasing market demand for more advanced services.

Over recent years, the object-oriented programming paradigm, with its new ideas of data abstraction, inheritance and encapsulation, has been widely adopted in software development. Because this paradigm also provides a good method for modeling complex distributed systems, the adoption of object-oriented approaches in distributing computing is natural and inevitable. This gave rise to the new but very promising area of Distributed Object Computing (DOC). As the confluence of two major areas of software technology: distributed computing systems and object-oriented design and programming [SCH97a], Distributed Object Computing technology not only provides an effective solution for the software integration over a heterogeneous environment [KIN96] but also simplifies the development of distributed software by alleviating the inherent heterogeneity and complexity of the underlying networks. This leads to distributed systems being of much more reusability, modularity, extensibility and location transparency. There is no doubt, the arrival of distributed object computing technology will also have great impact on distributed multimedia computing.

At the heart of the contemporary distributed object computing models are object request brokers (ORBs), which function as the communication bus between local and remote objects, thus effectively eliminating the tedious, error-prone and non-portable aspects of developing distributed applications, and enabling programmers to develop and deploy complex distributed applications rapidly and robustly [SCH97a]. Currently, two widely used ORB architectures are the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [OMG95], and the Distributed Component Object Model (DCOM) [CHU97], which is being developed by Microsoft. CORBA, as an industry standard, is being more and more adopted in the development of distributed systems.

2.2.3.1 OMG's CORBA

The goal and discipline of DOC is using Object-Oriented techniques to distribute reusable services and applications efficiently, flexibly, and robustly over heterogeneous computing and networking environment [SCH97a]. However, the inherent heterogeneity of the underlying networks is the big challenge. To overcome the heterogeneity, what has to be solved is the problem of application integration in a distributed heterogeneous environment [YAN95]. In recognition of this, the Object Management Group (OMG) was formed and there came out the CORBA architecture.

.• OMG

OMG was formed in 1989 with the main purpose of providing an approach to the problem of application integration [YAN95]. From the OMG's point of view, in the long run, application integration is more than the sharing of resources on the desktop, but rather it must encompass networking, programming languages, heterogeneous platforms and different implementation choices. In other words, the real application integration should enable an application to use and share another component regardless of the language in which that component is written, the type of operating system it is running on, regardless of its location in the network or the networking protocol employed [HOR96].

To obtain its goals, the OMG has created a reference architecture, called Object Management Architecture (OMA) which attempts to define, at a high level of abstraction, the various facilities necessary for Distributed Object Computing. The OMA is composed of an Object Model and a Reference Model.

• **OMA Object Model**

OMA Object Model [VIN97] [YAN95] defines common semantics for specifying the externally visible characteristics of objects in a standard implementation-independent way. In the OMA Object Model, an object, is defined as an identifiable, encapsulated entity and whose services can only be accessed through its well-defined interfaces.

An interface defines the attributes and the set of possible operations (i.e. services) that can be performed on an object. An operation (also called "method" or "signature") of an interface is an identifiable entity with a name, operational parameters and a return value. The implementation and location of the target object are hidden from the requesting object. Interfaces for objects are specified in the OMG Interface Definition Language (OMG IDL).

• **OMG IDL**

To allow for objects to be implemented using different programming languages and yet still be able to communicate with each other, language independent interfaces are important within heterogeneous systems [OMG95]. In OMG, object interfaces are described using OMG IDL.

OMG IDL is a formal declarative language with a syntax resembling that of C++. IDL provides a set of built-in types (such as long, float, double, and Boolean), constructed types (such as struct and union), template types (such as sequence and string) and an Object Reference type. These types are used to declare attributes types or parameter types and return types of the operation.

Since IDL is only used to define object interfaces, specifying their attributes and operations, there exists a mapping between the IDL code to the programming language which is actually used to implement the object. This work is done by the CORBA IDL

compiler. The IDL compiler accepts IDL code and generates client stubs and implementation skeletons, which are used for the target object invocation (we will explain it later). The current IDL mappings supported are C, C++, Ada and SmallTalk.

• **OMA Reference Model**

The OMA Reference Model identifies and characterizes the components, interfaces, and protocols that compose the OMA. The OMA reference Model is composed of four components as shown in the Figure 2.5 [OMG95].

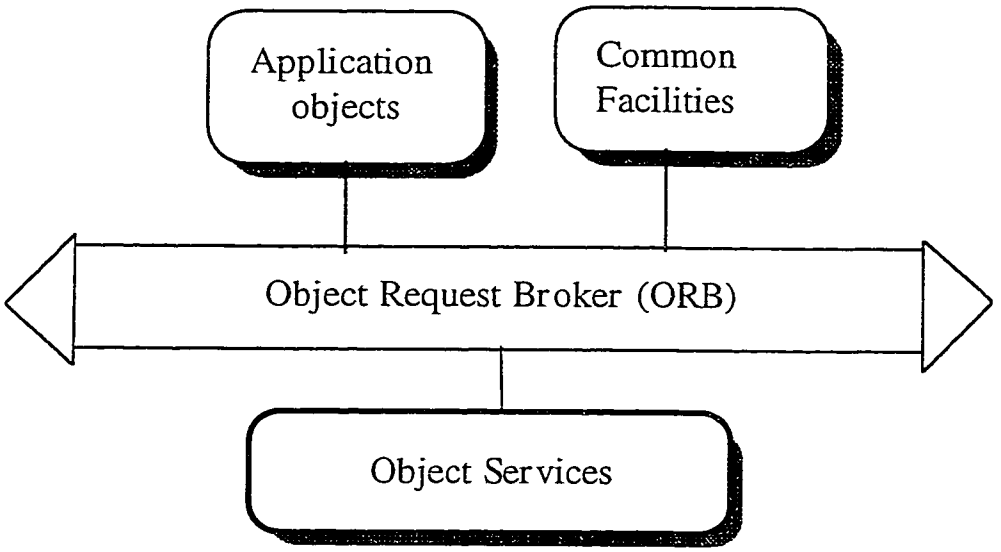


Figure 2.5 OMA Reference Model

At the heart of the OMA is the Object Request Broker (ORB), which acts as a communication bus between different objects, enabling objects to interact with each other in a distributed environment. The concrete description of this entity is defined in the "Common Object Broker: Architecture and Specification" document published by the OMG. It is normally referred to as CORBA.

The Object Services provide some low-level interfaces necessary for the application objects, such as Naming Service, Event Service, LifeCycle Service, etc. The

description of this entity is given in the "Common Object Services Specification" (COSS). It is also called CORBAservices [OMG94].

Common Facilities objects mostly provide some high-level services such as printing and email for the application objects.

• CORBA Architecture

In general, to simply the development and integration of distributed applications, CORBA provides a mechanism by which an object can make requests and get responses transparently despite the physical location of the target object [OMG95].

The CORBA architecture is shown in Figure 2.6.

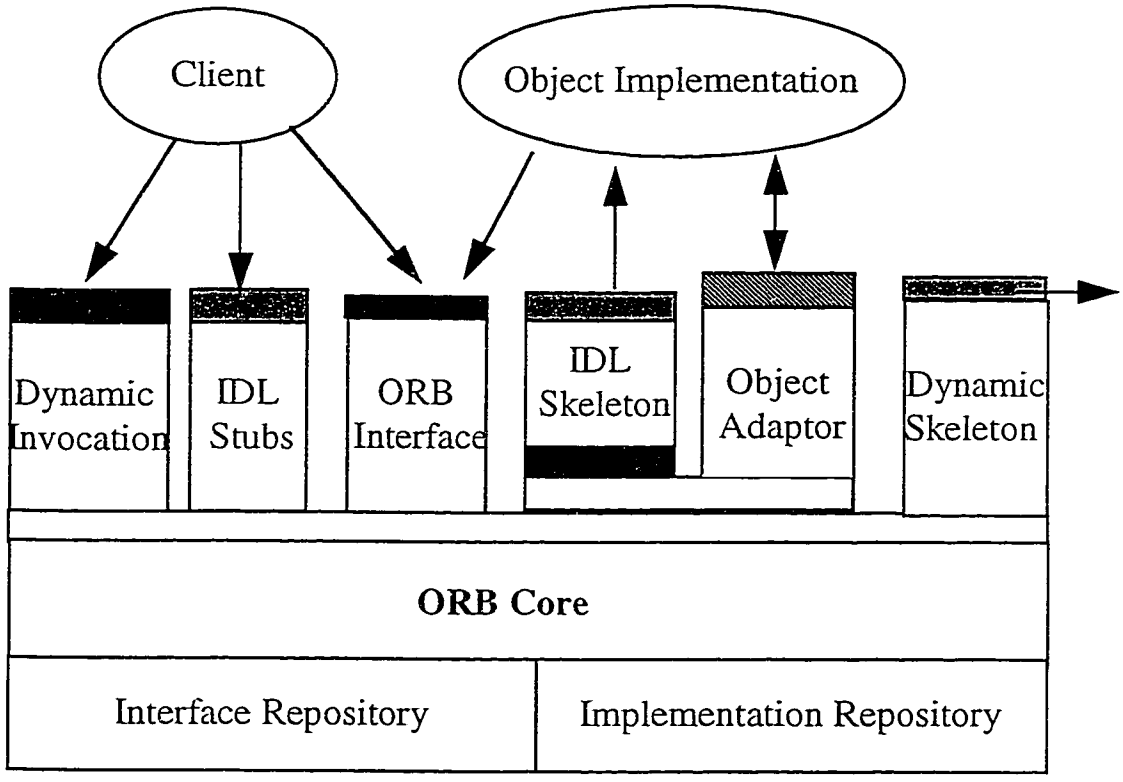


Figure 2.6 CORBA Architecture

The whole architecture can be divided into three parts: client-side interfaces, server-side interfaces and ORB core [YAN95] [SIE96].

In order for the client object to make requests and get responses, the ORB core plays the key role. It is responsible for accepting requests from the client-side interfaces, delivering the request to the server-side's object implementation and returning the results back. To enable the ORB core to locate the object's implementation, an Implementation Repository should be maintained.

On the client-side, when a client wants to use the service provided by another object, before it makes a request, it must first obtain the Object Reference of the target object. Actually, Object Reference is a proxy object. Any operations invoked on the proxy object will be forwarded to the server-side by the ORB core. To obtain an object reference, CORBA provided two ways: static invocation through the compiler-generated IDL stubs interface or dynamic invocation through DII (Dynamic Invocation Interface), which allows clients to create a request to the target object at run time. To enable the DII, the ORB must maintain an Interface Repository which stores the run time information about every registered IDL interface.

On the server-side, when the request arrives, the Object Adapter will be used to find the object's implementation and work together with the object skeleton to invoke the request on the object's implementation and then pass the results back to the ORB. Finally the results are returned back to the client by the ORB core.

2.3 CORBA-based Middleware

In short, CORBA provides some high-level facilities with which objects can transparently make requests and receive responses without taking care of the underlying complexity of the networks, thus it simplifies the task of developing distributed

applications. To deal with the complexity of multimedia communication systems and to facilitate the multimedia services creation, using CORBA in distributed multimedia computing is necessary. Concerning the multimedia networks, as mentioned above, they are composed of computer nodes equipped with multimedia devices interconnected via communication networks. The multimedia devices can be naturally considered and abstracted as objects. The interaction between them can also be supported by CORBA. Due to this, CORBA-based "middleware" has been extensively adopted.

As early as in 1992, with the arrival of the CORBA architecture, a "CORBA-based middleware" was first addressed by the Interactive Multimedia Association (IMA), jointly by HP, IBM and SunSoft, in its Multimedia System Services specification.

2.3.1 Multimedia System Services (MSS)

Multimedia System Services [IMA93] conceptually constitutes a framework of "middleware", i.e., system software components, lying in the region between the generic operating system and specific applications. Its primary goal is to provide an infrastructure for building computing platforms that support interactive multimedia applications dealing with synchronized, time-based media in a heterogeneous distributed environment.

Within such an infrastructure, based on object-oriented paradigm, multimedia devices and other resources are modeled as "objects". To enable multimedia computing in a heterogeneous distributed environment implies that multimedia device and media access is handled not only locally but also remotely. To support objects to interact with remote objects, the MSS relies upon the CORBA architecture.

As a middleware, the MSS is intended to marshal lower-level system resources to the task of supporting multimedia processing, and to provide a set of common services

which can be used by multimedia application developers on an industry-wide basis. The proposed middleware architecture can be described in Figure 2.7.

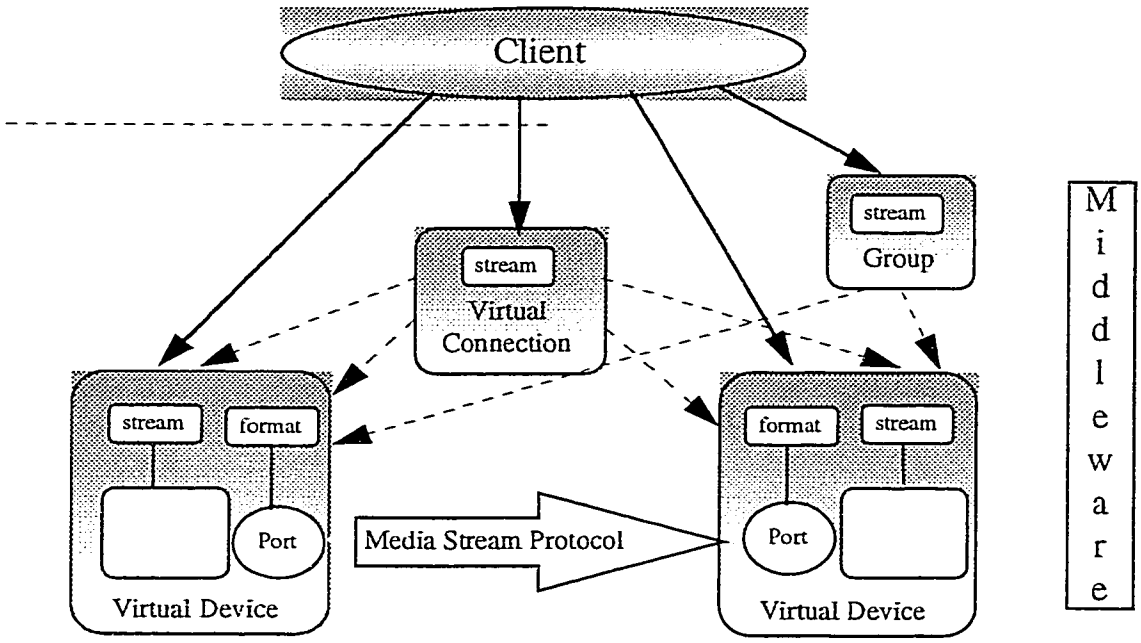


Figure 2.7 The MSS architecture

Each component in the diagram is considered as an object and specified in CORBA IDL. Each client in a MSS environment interacts with two virtual devices objects, which are bound by the Virtual Connection object. These components can run locally or remotely.

Each Virtual Device is a processing node. The nature of the processing (capture, encoding, filtering, etc.) varies according to the specific object. Associated with each virtual device is a stream object and one or more format objects.

A stream object provides the client with an interface to observe media stream position. Some stream objects also provide an interface for controlling the flow of data from and to the device. Some stream objects for controlling the synchronization between

streams are also provided. The Virtual Connections object may also have associated stream objects to control the media flow.

In addition to a stream, a virtual device also contains one or more ports, describing an input or output mechanism for the virtual device.

The virtual connection provides an interface to create a connection between an output port of one virtual device and an input port of another, fully encapsulating low-level transport semantics. Two classes of virtual connections are provided: unicast and multicast.

The group object is used to assist the client object to manage the two virtual devices and the virtual connection. It also provides an effective mechanism for atomic resource allocation and control end-to-end QoS.

An interface-inheritance diagram for the CORBA IDL interfaces is defined in Figure 2.8.

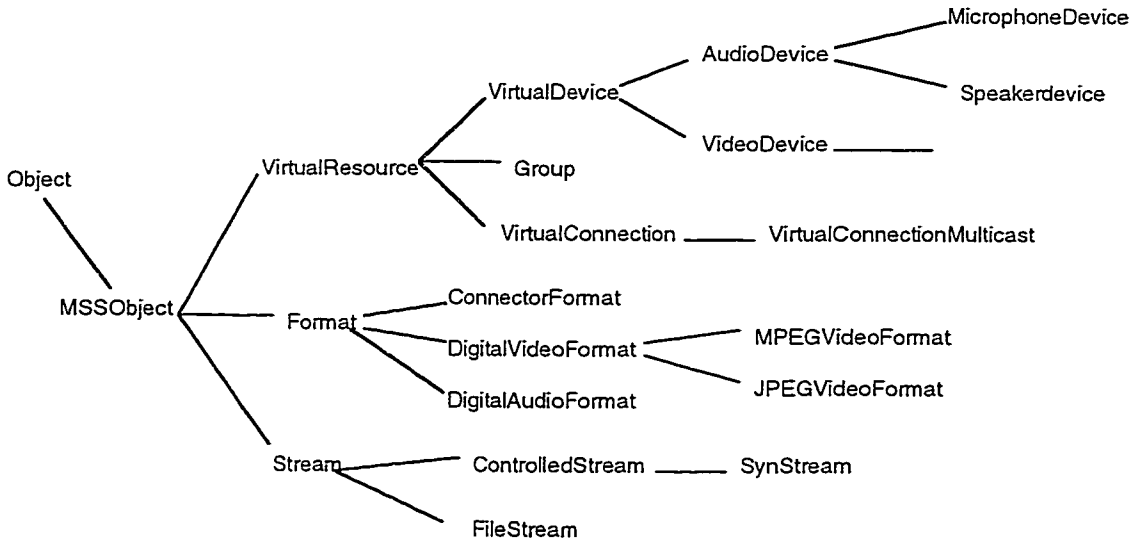


Figure 2.8 MSS interface inheritance diagram

2.3.2 Related Work

Generally, the MSS middleware provides a standard CORBA-interface to the resources in computing platforms, networks and multimedia devices. CORBA provides an interoperable control protocol for object interaction. Due to the object-oriented nature of CORBA, such architecture is modular and extensible, thus the service creation and deployment in heterogeneous environments is much flexible.

Centering around the MSS architecture, a group from the National University of Singapore, has designed and implemented a Multi-party Conference System over ATM networks in 1995 [CHO95].

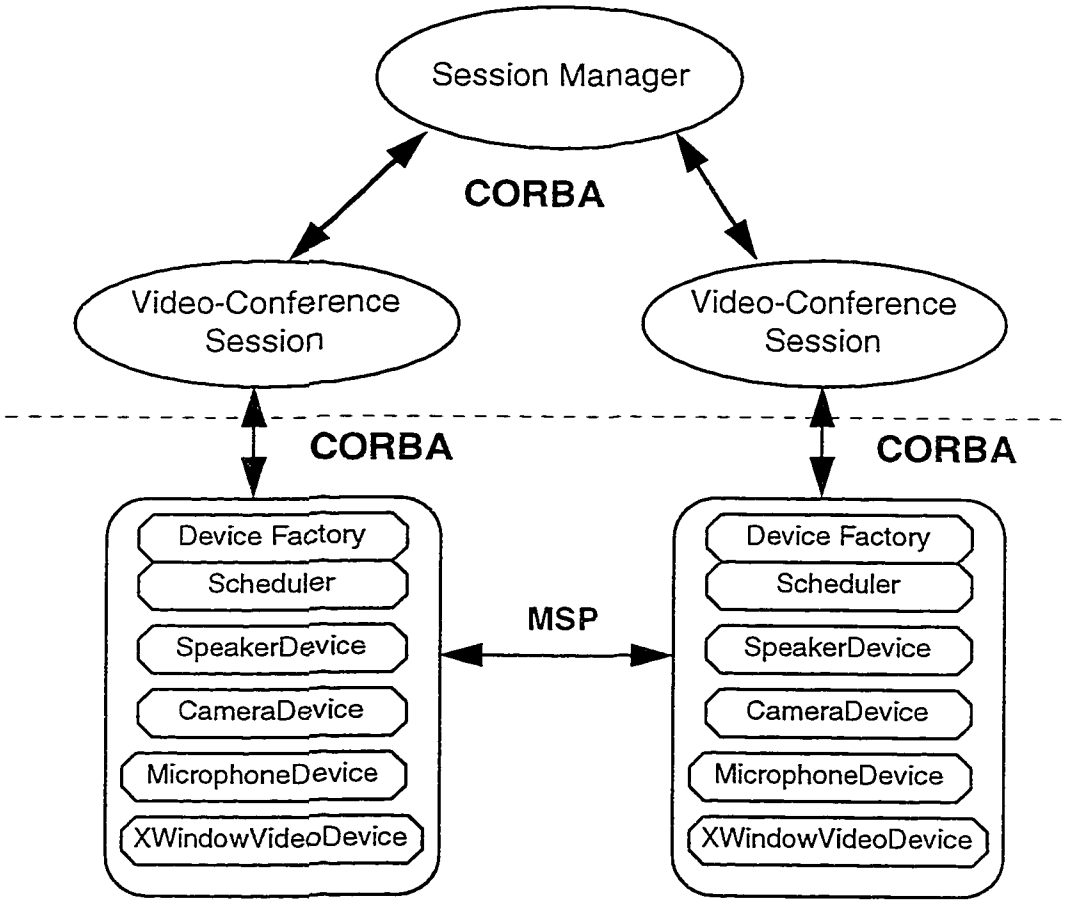


Figure 2.9 Multi-party video conferencing system architecture

They implemented most of the interfaces in the MSS on the SPARC platform running Solaris 2.3. For example, a Virtual Device provides an interface for media-processing, including microphones, cameras and etc. The device also has an associated Stream and Format for each Port, which are used for managing the media stream and media format, respectively. The Media Stream Protocol handles media transport between devices. In order to setup connection between devices, the VirtualConnectionUdp and VirtualConnectionFore interfaces have been defined to provide unicast connections through UDP and AAL, respectively. Moreover, VirtualConnectionMudp and VirtualConnectionMfore interfaces are defined to provide multicast connection.

On the top of the middleware, a service architecture is also defined for developing distributed interactive multimedia services. The whole software architecture of Multiparty Conference System is shown in Figure 2.9.

In addition to the work introduced above, a "middleware" employed in a" general-purpose distributed multimedia system", named MAESTRO, developed by Pohang University, Korea, could be another good example.

In MAESTRO [YUN95a] [YUN95b], a distributed multimedia system is conceived as a layered architecture, which can be described in Figure 2.10.

At the top layer, there exist various multimedia applications, such as video conferencing, video-on-demand, whiteboard, and etc., which use the services provided by the second layer.

The second layer provides distributed multimedia services used by the top layer's multimedia applications. This layer is designed using the object-oriented approach and consists of objects which can be used to transfer multimedia data and to abstract devices. This objects are tailored to form a distributed application.

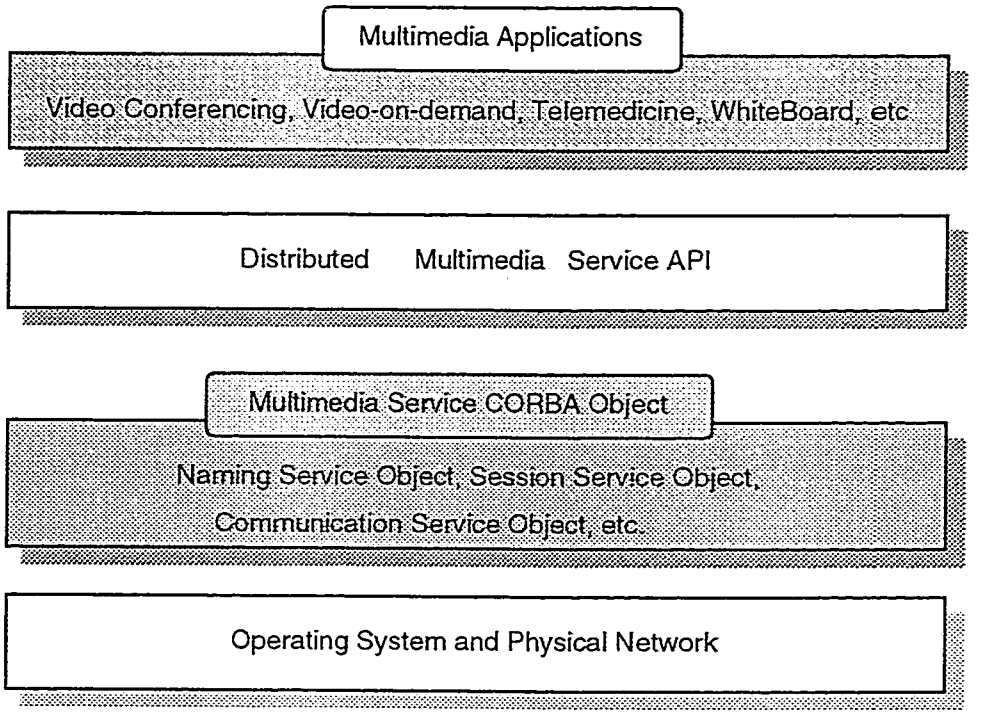


Figure 2.10 MAESTRO architecture

The third layer consists of various CORBA objects that implement the services required to support multimedia applications, including Naming Service Object (NSO), Communication Service Object (CSO), Storage and Retrieval Service Object (SRSO), Session Service Object (SSO), and Management Service Object (MSO). Actually these CORBA objects are used to provide run-time support for higher-level applications, such as exchanging multimedia data, storing and retrieving data and so on. The interaction between the different CORBA objects is through the use of CORBA standard.

At the lowest layer, these exist various networking technologies, such as FDDI, ATM, SONET, Fast Ethernet, etc. with various hardware and operating system platforms, such as Solaris, SunOs, HP-UX, and etc.

Obviously, layer2, together with layer3, constitutes what we called "middleware".

All in all, employing CORBA-based middleware into the distributed multimedia system, the complexity of the architecture can be greatly simplified and the service creation and deployment becomes much more flexible and easier. However, as far as the object interaction is concerned, what CORBA can support is only limited to the simple object's making requests and getting responses. Currently CORBA does not support real-time interaction nor continuous transfer of isochronous data stream between objects, while the transfer of continuous data can not be adequately expressed in terms of object invocation.

From this point of view, CORBA does not adequately satisfy the more complicated and stringent requirements of distributed multimedia applications. Therefore, efforts are made for real-time CORBA by extending the current ORBs to better support time constraints multimedia applications. These include research for "High-performance End System Architecture for Real-time CORBA" from Washington University [SCH97b], the DIMMA (Distributed Interactive Multimedia Architecture) project being conducted by APM-ANSA Laboratories, and the MMN (Management of Multiservice Networks) by the British Telecom University Research Initiative (BT-URI) [SIQ97].

Chapter 3

Programming the Future Information Networks

3.1 Information Networks

In recent years, due to the deregulation of its markets and the extensive deployment of broadband communication technologies, the telecommunications industry is facing new challenges. The market deregulation is opening up considerable opportunities for the provision of newer and more advanced telecommunication services [CHA95a] [STEF95]. As a consequence, the network operators are currently experiencing a "new world" of service competition. Rapid and timely new services delivery in response to the

market demand becomes a key factor in determining the success of future telecommunication service providers [LAZ97].

On the other hand, with the deployment of the World Wide Web on the Internet, the computer-based information services market is growing rapidly. This implies that the telecommunication industry can no longer be in isolation from the computer industry. If the future telecommunication industry restricts itself to the telephony area, the control of the services market will be out of their hands [STEF95]. Meanwhile, without help from the telecommunication industry, it is hard for the computer community to provide services on a global basis. Therefore, the convergence of the telecommunication industry and computer industry is in great need. This gave rise to a new vision of "information networks".

It is envisioned that an information network should [CHA95a] [BAR93] [NAT92]:

- Offer a rich variety of information services, including conventional telephony, and data transmission services, as well as sophisticated multimedia services;
- Be manageable by different stakeholders such as service providers, network operators, end users and customers;
- Provide networks' users the capabilities of on-demand access and management of information any time, any place, and in any form.

This vision implies that the software architecture of such information networks should be open and flexible enough to allow for the timely and easy introduction of new services. The software architecture is, by necessity, constructed as an open distributed system, taking advantage of the recent advances in the distributed computing and object-oriented analysis and design and other concepts and standards from the telecommunications as well as the computer industry, such as ODP (Open Distributed

Processing), CORBA, and so on [DUP94]. The Open Distributed Processing Reference Model (RM-ODP) constitutes the basis of the information network architecture [BER95] [STEF95].

3.1.1 ODP Reference Model

The Reference Model for Open Distributed Processing [ODP95], being standardized by the International Standard Organization (ISO), aims at defining a framework for open distributed systems.

The fundamental concepts of the reference model are "objects" and "interfaces". Objects are considered as units of distribution. The interfaces of an object correspond to access points to it. An object may have several interfaces. The interaction among the objects only occurs at the interfaces.

In the ODP Reference Model, an open distributed system can be "viewed" from five aspects, called "viewpoints". Each viewpoint corresponds to a particular perspective, allowing a system to be concerned from a particular angle. The five viewpoints are [BER95] [MAGE93]:

- Enterprise: a viewpoint of an ODP system that focuses on the purpose, scope and policies of the system;
- Information: a viewpoint of an ODP system that focuses on the semantics of information and information processing activities in the system;
- Computational: a viewpoint of an ODP system that enables distribution through functional decomposition of the system into objects which interact at interfaces.
- Engineering: a viewpoint of an ODP system that focuses on the mechanisms and functions required to support distributed interaction between objects in the system.

- Technology: a viewpoint of an ODP system that focuses on the technologies to build the systems.

For each viewpoint, some concepts and rules are also defined in the ODP Reference Model for the specification of the ODP system, which are called viewpoint languages. In general, the computational and engineering languages encompass the minimum requirements for building an open distributed system.

In the computational languages, the open distributed system is composed of a collection of interacting computational objects with well-defined interfaces. The communication pattern among the objects can be either in the form of implicit binding or explicit binding. Implicit binding refers to the basic inter-object communication pattern of the operation invocation, which is usually supported by the infrastructure, such as CORBA. Explicit binding results in the creation of a special computational object, called binding object, which supports a binding between a set of other computational objects [BER95]. Binding in general denotes a set of actions or processes for associating or interconnecting different components (objects) of a system. The advantage of the explicit binding is that it makes the explicit control of associating and interconnecting different objects of the system possible.

In the engineering languages, an ODP system consists of a set of interconnected processing nodes. A node refers to an abstraction of information processing resource together with its basic system software. A computer and the software it supports is an example of a node. Each node runs a nucleus, called DPE kernel, which will be discussed later. The set of cooperating nuclei constitutes a distributed platform that supports the execution of applications, which is usually called Distributed Processing Environment (DPE) [CHA95b].

With the ODP Reference Model as the basis, let's see what the information networks look like.

3.1.2 Structure of Information Networks

Simply, from the ODP engineering point of view, the structure of the information networks can be understood as comprising a set of interconnected processing nodes. The processing nodes can be arbitrary machines, from network elements of the transport network (e.g. switches, hubs, routers, etc.) to standard PCs, workstations that lie at the periphery of the transport network. The communication between the processing nodes is mainly akin to the message exchanging between the computational objects on each processing node. From this perspective, the way that the processing nodes are connected denotes a "virtual network", called "kernel transport network". The kernel transport

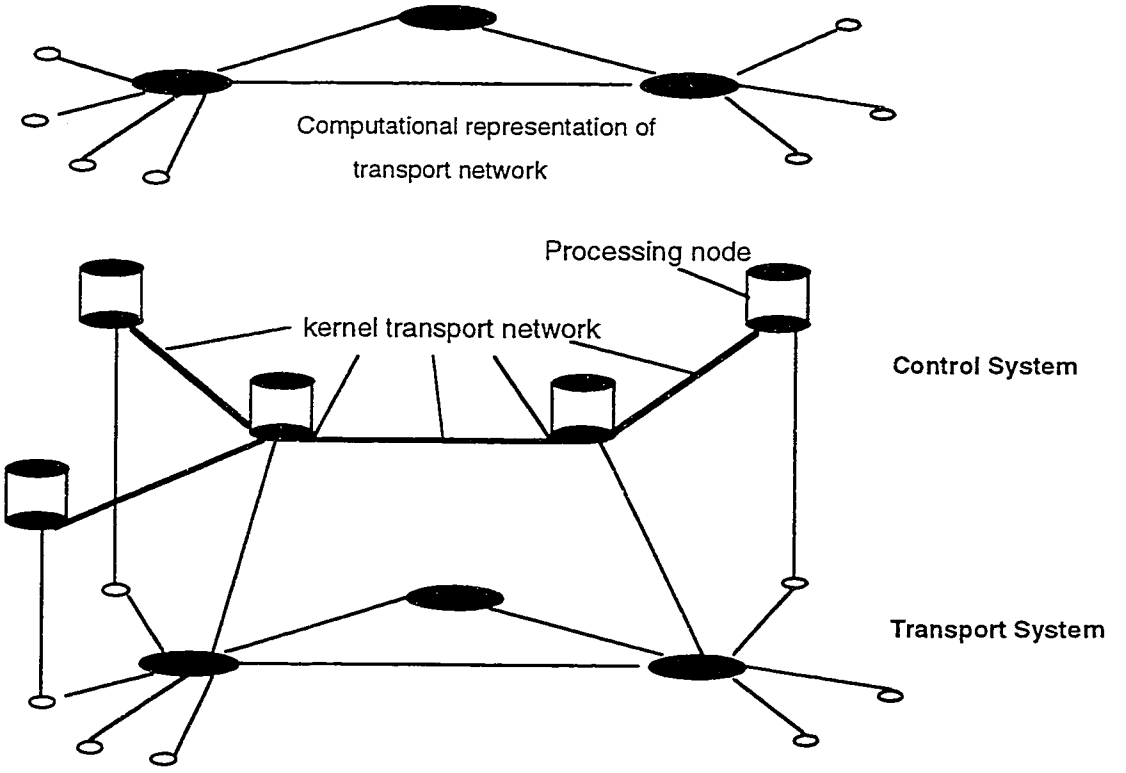


Figure 3.1 Structure of Information Network

network is logically different from the transport network in the sense that the "virtual" kernel transport network is required for the interconnection of processing nodes, while the transport network is used for the transmission of data. Regardless of the logical difference, essentially the kernel transport network and the transport network are implemented by the same physical network. The structure of the information network can be illustrated in Figure 3.1.

As far as the information network services are concerned, from the ODP computational point of view, they are constructed as distributed applications being composed of a set of interacting objects. The objects, representing the different software components, may distributively run on the various processing nodes. The object may be arbitrary, which can either denote a physical entity or a well-defined functionality. Therefore, if the minimum access to the elements of the transport network is available, i.e., the control and management of them are given by their object interfaces, the primitive information network services for the resource control and management of the transport network can be simply performed by associating and interconnecting (called binding) different transport network objects. Meanwhile, these primitive information network services, such as connection management, can also be identified and manipulated as individual objects. As a result, based on the basic network services for the resource control and management of the transport network, the high-level services creation can be carried out via a high-level programming paradigm. This leads to the most important feature of the information network, i.e. programmability. This overcomes the limitation of the UNI/NNI architecture, thus greatly facilitates the easy services creation within it.

The Telecommunication Information Networking Architecture (TINA), currently under development by the TINA Consortium (TINA-C), is a good attempt at a software architecture for the future information networks.

3.2 TINA Architecture

As mentioned before, the major trends in the telecommunication society, such as the increased service competition and the explosive increase of the services variety, point to an urgent need for an information network architecture in which services and their management can be introduced easily, quickly and smoothly. Such an architecture is being tackled by the TINA-C's TINA.

3.2.1 Layered View of TINA

The Telecommunication Information Networking Architecture Consortium (TINA-C) is an international consortium of telecommunication operators, computer manufactures and telecommunication vendors, aiming at defining an "open" software architecture for future information services and market in which different stakeholders, such as customers, end users, service providers, and network providers, are involved. The design principle of the TINA architecture is based on the adoption of ISO Reference Model for Open Distributed Processing (RM-ODP) and the distributed object computing technology, such as CORBA, where CORBA is used to address its core element, i.e. Distributed Processing Environment (DPE) [APP93] [KIN96].

In addition to enabling the rapid introduction of information networking services, which include not only traditional voice-based services but also advanced multimedia services, the TINA architecture is also intended to be independent from any particular underlying technology. This is performed by integrating the network and operations applications via a common distributed processing platform, which hides from applications the effects and complexities introduced by distribution. The switching and transmission equipments of transport network are specified as managed objects with generic interfaces [NAT92].

The basic layered architecture of TINA [CHA95b] can be described as in Figure 3.2.

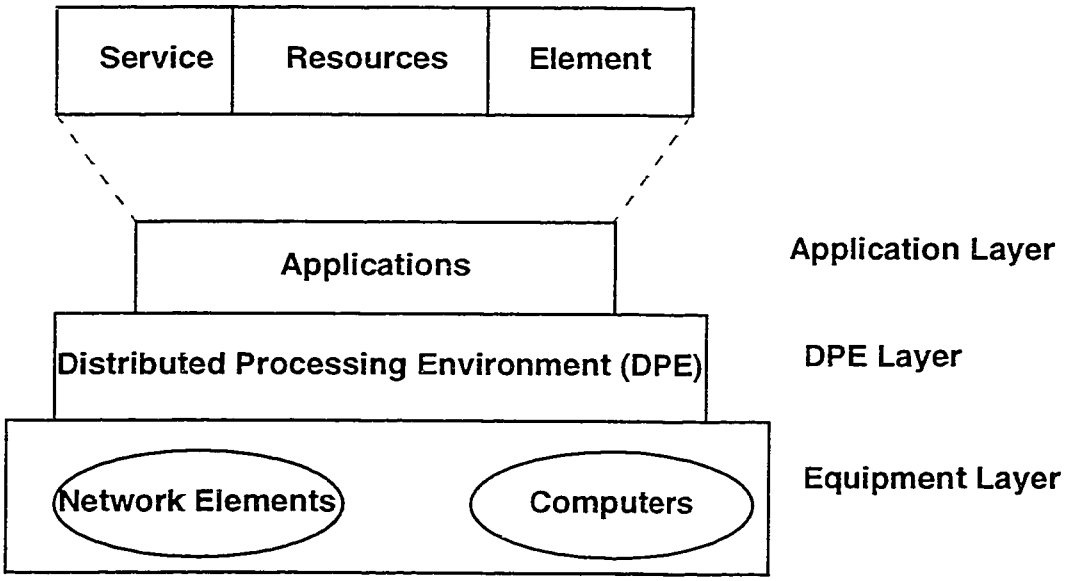


Figure 3.2 Layered TINA Architecture

As shown above, at the bottom is the equipment layer corresponding to the physical devices in the system, consisting of computers and transport network elements. In the middle is the Distributed Processing Environment (DPE). The DPE layer provides a generic computing platform, which hides the complexity and distributed nature of the system in the networks. Since OMG's CORBA functions exactly the same as DPE, it is being considered as an good candidate implementation for the DPE. On the top of the DPE is the software application layer partitioned into three sublayers: service, resources and elements.

In the telecommunication information networks, the transport network provides a set of switching and transmission resources that may be used and managed by software in the application layer. The software applications in the application layer mainly concern themselves with the provision and operation of services, as well as the operation of the transport networks.

Specifically, the element sublayer is composed of objects representing atomic units of physical or logical resources, defined for resource allocation, usage and management purpose. Objects in the element layer are called elements, which can be considered as the abstract representation for the physical equipment found in the transport networks, such as switch fabrics and transmission equipment. As such, the resource allocation and management can be done via the associating or binding of the elements.

The resources sublayer contains objects that can manipulate collections of objects in the element layer and provide the service layer with abstract representation of elements. The objects in the resources layer mainly contain the managing functions to be applied to elements, such as connection management. In general, the resources layer is responsible for providing technology independent views of elements, so as to be suitable for high-level services creation.

The service sublayer consists of objects involved in the provision of more services to stakeholders. The delivery of new services may use services provided by the service layer or the other two layers.

The interaction between the objects is supported by the DPE. As we can see from the above layered TINA architecture, with the support of DPE, the operation of the transport network and the high-level services creation can be performed in a consistent and flexible manner.

In accordance with the layered architecture that is described above, a set of concepts and principles is specified, with which the TINA architecture is divided into three subsets: logical framework architecture, service architecture and management architecture [CHA95a].

3.2.2 Logical Framework Architecture

The logical framework architecture of TINA is largely based on the ODP. As we mentioned before, for developing open object-oriented distributed applications, five viewpoints are given in the RM-ODP, including information, computational, engineering, enterprise and technology viewpoints. Only the former four of them are considered in TINA, which are information, computational and engineering and enterprise. For each of the viewpoints, a set of modeling concepts is defined [CHA95b]. Among these, the computational and engineering modeling concepts are of our interest.

The computational modeling concepts provide a framework focusing on the distributable software objects and their interactions. The distributed applications are composed of computational objects. The computational objects are the units of programming and encapsulation. Computational objects interact with each other to provide the application. Objects interact with each other by sending requests and receiving information to and from well-defined interfaces. In TINA, there are two forms of interfaces that an object may offer or use: operational interface and stream interface. The notation for computational specifications is TINA-C ODL (Object Definition Language), which is an enhancement of OMG IDL to allow the definition of objects that may have multiple interfaces.

The engineering modeling concepts provide a framework focusing on how the computational objects are distributed among the nodes of the system.. To support their execution and interaction, an infrastructure is also defined. Such an infrastructure is called Distributed Processing Environment (DPE). The structure of the DPE is illustrated in Figure 3.3

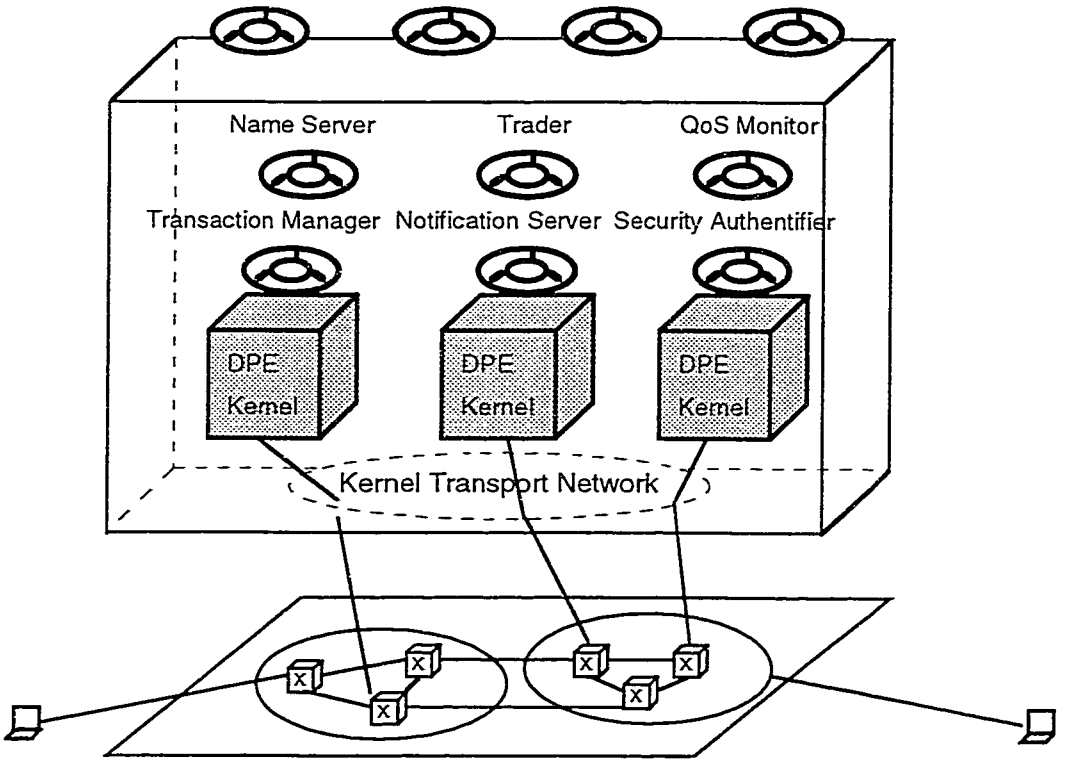


Figure 3.3 DPE structure

From the application designer's point of view, the DPE can be considered as one homogeneous infrastructure hiding the complexity and heterogeneity of the underlying network and network resources, and providing the mechanism that allows objects to interact with one another without depending on the nodes they are on [CHA95a] [DUP94]. To obtain its objective, the DPE consists of DPE kernels implemented on top of different heterogeneous computing environments, interconnected by a specific logical network, i.e., Kernel Transport Network (KTN) that we mentioned before. The DPE kernel provides support for the object-life-cycle control and inter-object communication. The object-life-cycle control corresponds to the capabilities of creating and deleting objects at run-time. Inter-object communication provides the mechanisms to support the invocation of operations provided by interfaces of remote objects. A DPE kernel is assumed to be present on all nodes that contain a DPE. From this perspective, the DPE kernel actually functions the same as Object Request Broker of OMG's CORBA.

Additionally, DPE kernels are enhanced by DPE servers, which provide services dealing with the run-time execution and communication of objects, such as trading services, notification services, and so on. Similarly, the DPE services function the same as the CORBAservices. Therefore, CORBA has been considered as a good candidate for the DPE implementation.

With the support of the DPE, and in conjunction with the layered TINA architecture, as mentioned before, we may see that if the generic interface to the technology dependent transport network resources are provided by the element layer, upon the logical kernel transport network, the operation, control and management of the transport network can be performed in a very flexible and technology independent manner. That is, it allows different authorities to manage and operate them, even using different policies, and the different realizations can be chosen. In turn, based on the basic services for operation and management of the transport network, other high-level services can be conducted easily by interconnecting a set of computational objects representing the basic transport network services. This denotes the TINA principle of separation services from communication control and management. The separation of services and communication allows for easy introduction of new services independent of the underlying network.

To make the information network services more versatile, easier to develop, more timely to delivery, interoperable, and independent of the underlying specific network and computing environment, TINA service architecture is also defined by TINA-C.

3.2.3 Service Architecture

The TINA Service Architecture [BER95] [TIN95] defines concepts and basic principles necessary for the construction of TINA services. In TINA-C, from the ODP enterprise

viewpoint, a TINA service is defined as "a meaningful set of capabilities provided by an existing or intended network to all who utilize it, like customers, end users, network providers and service providers". From the ODP computational viewpoint, a service can be considered as a set of capabilities provided (or offered) by a computational object that can be used by other objects through its interfaces.

The objectives of the TINA service architecture can be outlined below [TIN95]:

- Support of a wide range of services. The service architecture is intended to support a wide variety of services, including communication services, management services, information services, etc. The communication services are responsible for the establishment of connections. Management services are responsible for the management of TINA resources. In addition, the architecture should be open to allow the introduction of new classes of services.

- Rapid service development and provision. The service architecture must support the rapid development and deployment of services in order to respond promptly to market needs and at a less cost.

- Tailored services. TINA services must be easily tailored to satisfy specific requirements of a variety of customers. Moreover, services subscribers and end-users should be offered some direct control in managing their services.

- Independent evolution of services and network resources. Services should be defined independently from a specific network technology. This facilitates the rapid introduction of new services and swift modifications of existing ones.

- Support for an open environment. The service architecture should fit in a multi-supplier/provider/operator environment, in which the introduction and modification of services from different vendors and authorities could be possible.

To obtain the objectives that are mentioned above, there are four concepts defined in the service architecture: session, access, management and resources. Among these, resource concepts define the objects and interfaces that provide service abstractions of transport network elements. Management concepts address the service management aspects.

Session concepts define those objects and interfaces that are required to support the initiation and termination of services. The term "session" refers to the temporal period during which activities are carried out. Three types of sessions have been identified: service session, user session and communication session. The service is activated by the service session, which is computationally represented by a service session manager. The user session represents the user's interaction with a service session, which maintains the state about a user's activities. The communication session is a

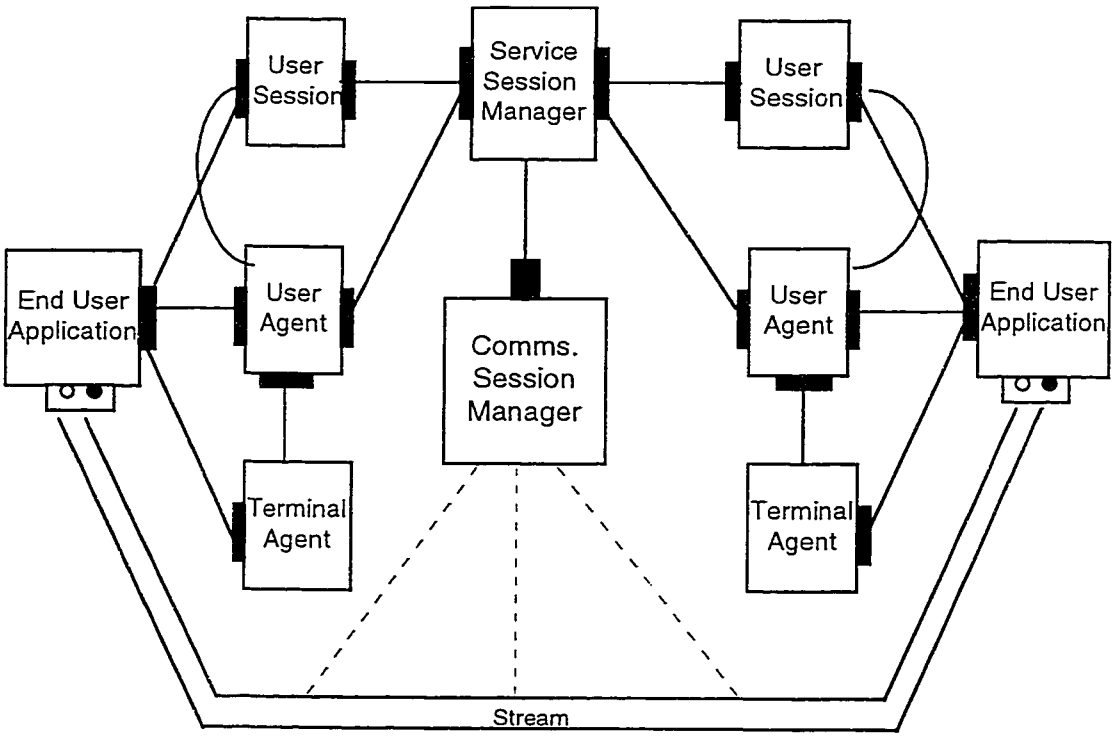


Figure 3.4 TINA Service Architecture

service-oriented abstraction of connections in the transport network, which is computationally represented by the communication session manager. The communication session maintains the state about the connections of a particular service session. The communication session provides a connection interface for a service session manager to manipulate. When the service session manager requests a connection between objects, the communication session manager in turn calls upon connection management objects to establish physical connections between the computing nodes (see Figure 3.4).

For the interest of users, they need to have flexible access to services, in terms of the locations from which they access the service and the type of terminals they use. The agent concept is used to define the access model. An agent is a computational object that acts on another entity. Two types of agents: the user agent and the terminal agent, are specified. The user agent is a computational object that represents and acts on behalf of a user. It receives requests from users to establish service sessions, or to join existing service sessions. A terminal agent is a computational object representing a terminal. In order to access a service, users must associate their user agents with terminal agents (see Figure 3.4).

In short, the TINA service architecture separates access session and service session, and the service session may then use communication sessions. This separation of services and communication sessions allows easy introduction of new services independent of the underlying network. This is crucial for the whole TINA architecture. And also with the support of DPE, the whole process of service creation can be conducted on a high-level programming paradigm.

As a proof-of-concept of such a new network software architecture, and also with the objective of programming multimedia networks, an open programmable environment

for multimedia networks, called "Xbind", was developed by the COMET group at Columbia University.

3.3 Xbind

Xbind [LAZ96a] [CHAN96], also called a "broadband kernel", is an open programmable operating platform that supports the creation, deployment and management of networked multimedia services on ATM-based broadband networks with end-to-end QoS guarantees.

Its principle goal is to provide an open programming environment that facilitates the easy creation of distributed multimedia services. Open means that the architecture must support functional Application Programming Interfaces (APIs) for resource control and management that other service providers can use for developing more useful services. Programmable means that these APIs should be "high-level' enough to allow the service creation to be carried out via a high-level programming language.

In reality, the research on the binding architecture of the Xbind "broadband kernel" is incorporated with the G-model of the XRM (eXtended integrated Reference Model) architecture [LAZ95b] [LAZ94].

3.3.1 Extended Integrated Reference Model (XRM)

XRM models the communications architecture of broadband networks and multimedia computing platforms in a consistent and integrated way. It is based on the observation that both the network and multimedia resources can be modeled as producers, consumers and processors of media, and the only difference lies in the overall goal that a group of devices is set to achieve in the network or the multimedia platform [LAZ95b]. The XRM architecture is given in Figure 3.5.

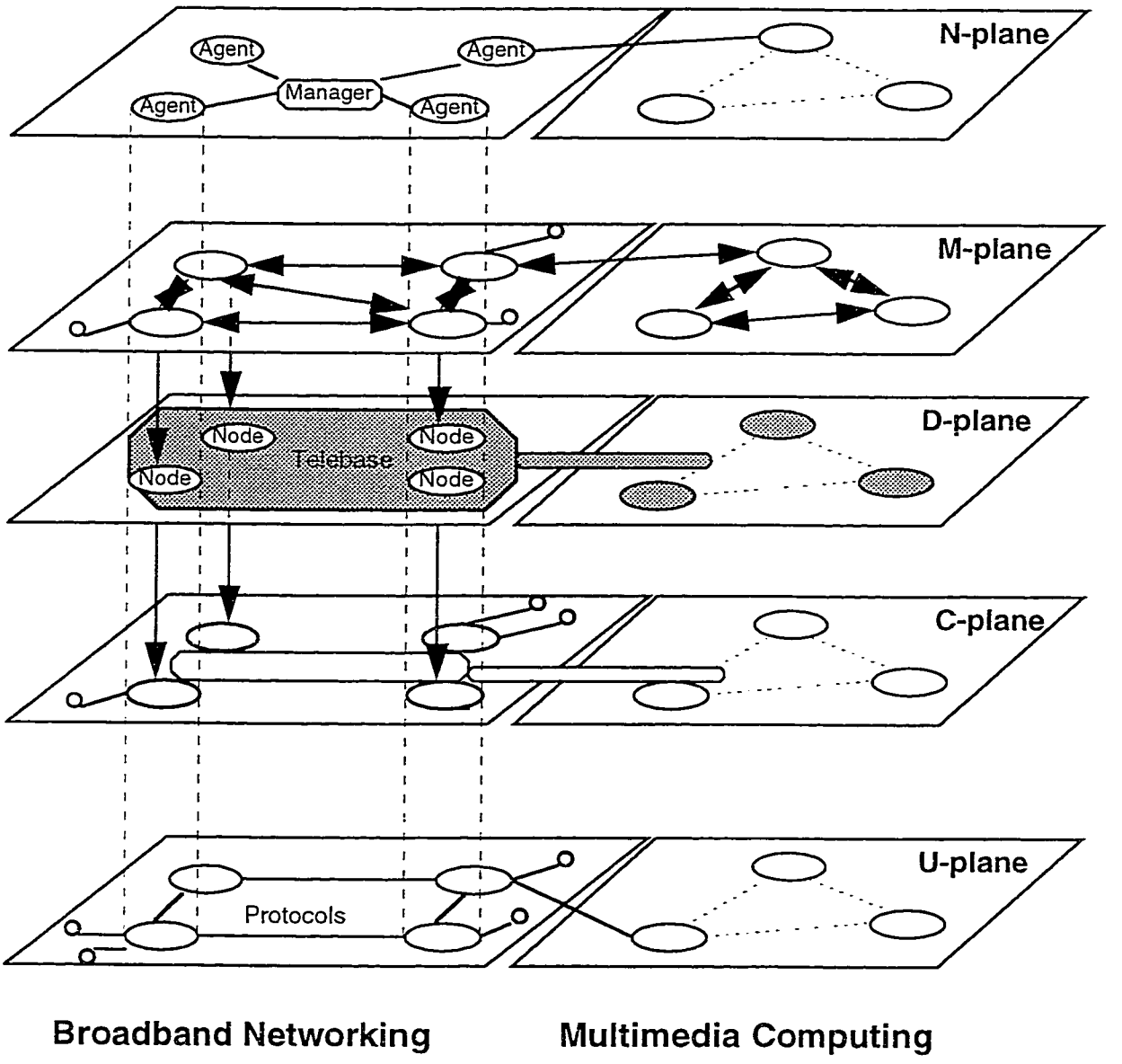


Figure 3.5 XRM architecture

Generally, from either the broadband networking or the multimedia computing side, the model is organized into five planes: N-plane for the network and multimedia system management; M-plane for the resource control, such as admission control, flow control, etc; C-plane for the connection management or binding; U-plane for transport of user information within the Customer Premises Equipment; D-plane contains objects modeling the network entities or multimedia devices.

In order to make the structure of XRM more transparent, three submodels within in the XRM are identified, which are called the R-, the G- and the B-model, respectively.

3.3.2 RGB Decomposition of The XRM

Based on the RGB decomposition, the communications architecture of networking and multimedia computing platforms is composed of three components, which are the Broadband Network (B- Model), the Multimedia Network (G- Model) and the Service and Applications network (R- Model). (see Figure 3.6) [LAZ95a] [LAZ95b].

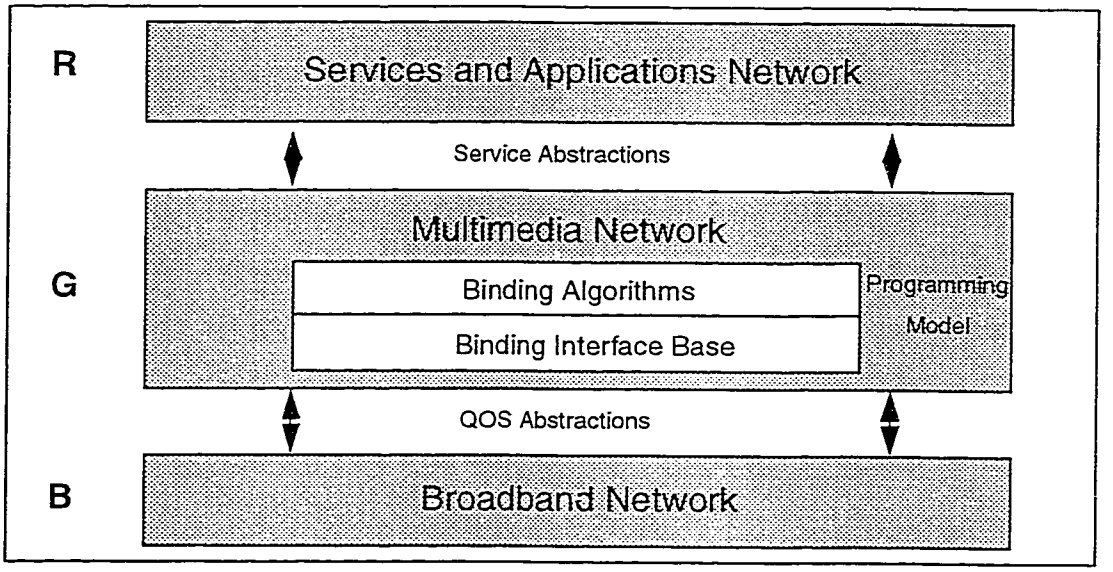


Figure 3.6 RGB Decomposition of the XRM architecture

The broadband network is defined as the physical network that consists of switching and communication equipment and multimedia devices. Upon this physical layer resides the multimedia network, which functions as a middleware to provide a set of basic network service for the resource control and management to the Service and Applications Network. The construction of these services by the multimedia network is based on the QoS abstractions provided by the Broadband Network. The QoS abstractions, by themselves are a passive representation of resource states. Similar to the TINA architecture, based on the abstraction of resources, the process of the basic

network service creation for resources allocation and management can be simply described as the states manipulation (or binding) through the use of binding algorithms. On the other hand, the services creation in the upper Service and Application network can also be simply carried out through dynamic composition and binding objects provided by the multimedia network. From this point of view, the multimedia network provides an open programming environment, based on which the multimedia service creation can be highly facilitated.

3.3.3 Binding Model and Binding Architecture

At the heart of Xbind broadband kernel is a conceptual framework, called the Binding Model, which consists of two building blocks: the Binding Interface Base (BIB) and a set of binding algorithms running on top of it. "Binding" refers to the activity of associating different components of a multimedia system, including not only network entities but also multimedia devices, to create a requested multimedia service with QoS guarantees. As such, a seamless binding environment between the network and multimedia resources is achieved.

The formal description of the Binding model is given by the binding architecture. The whole architecture is intended to be open, in which all multimedia networking entities, including switches, links, multimedia devices, participating in the binding process are modeled as computational objects with well-defined interfaces. All the interfaces reside in a data repository, i.e., Binding Interface Base (BIB) [LAZ96e]. The communication among these objects is supported by CORBA. The scalability of the binding architecture is readily achieved by simply adding new interfaces into BIB.

The interfaces in the BIB provide open and uniform accesses to abstractions that model the local states of networking resources. Binding algorithms play key roles in the service creation through the process of binding network entities and multimedia devices.

In essence, the BIB interfaces provide some kind of "software wrappers" around physical hardware devices so that they appear as software components. The components can then be managed and controlled by other software components to build services which may also be utilized by other components [LAZ95a].

The binding model gives the XRM:G its operational capability through service creation. In this model, service creation is defined as a process of object composition whereby objects are manipulated by algorithms residing in various XRM:G planes. More specifically, BIB resides in D- plane and the binding algorithms execute from within the M- and C- planes, which can be illustrated in Figure 3.7.

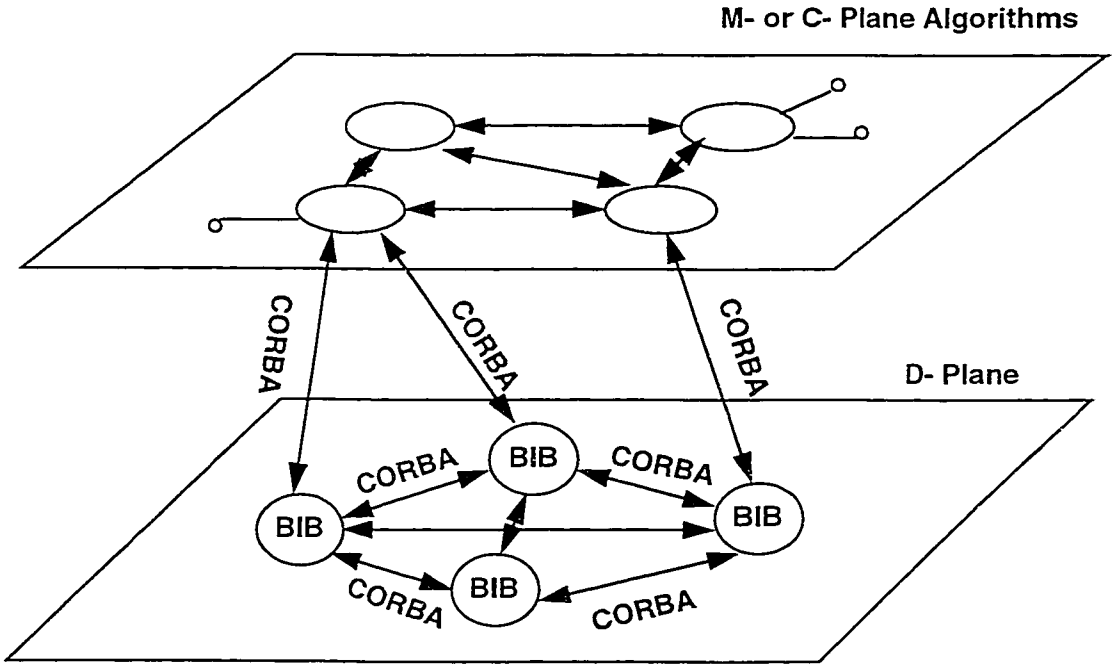


Figure 3.7 Network View of the Service Creation Process

3.3.3.1 Binding Interface Base (BIB)

All the interfaces in the BIB are defined using CORBA's IDL (Interface Definition language), and organized as the following inheritance hierarchy tree with restriction to the ATM networks (see Figure 3.8).

The key interface in the BIB is the VirtualResource. The inherited interfaces from VirtualResource model all the physical resources that are present in the networked multimedia computing. The VirtualResource is subdivided into MediaProcessor and MediaTransporter. The MediaProcessor models all the producers, consumers and processors of media, such as VirtualCPU and VirtualDevice. Similarly to the MSS, The VirtualDevice abstracts the functionality of a multimedia device, like camera, speaker, etc.

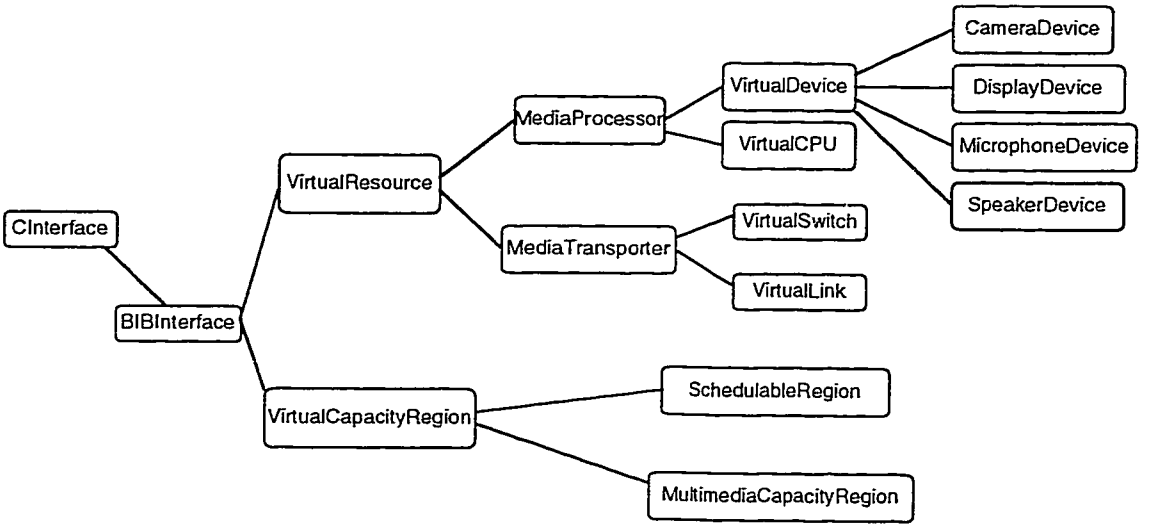


Figure 3.8 The Binding Interface Base

Different to the MSS, the network entities are modeled by MediaTransporter, and subclassified into VirtualSwitch and VirtualLink. The VirtualSwitch controls the manner in which VPI/VCI pairs are allocated and deallocated in the switch fabric part of a switch node. The VirtualLink models the cell and call resources consisting of output buffers and links. Such that a switch node is modeled as shown in Figure 3.9.

In general, the two interfaces of MediaProcessor and MediaTransporter effectively model the multimedia computing entities. The relationship between them can be shown in Figure 3.10. Compared with MSS, it it obvious to notice that the distinction between the MediaProcessor and MediaTransporter is not considered in MSS.

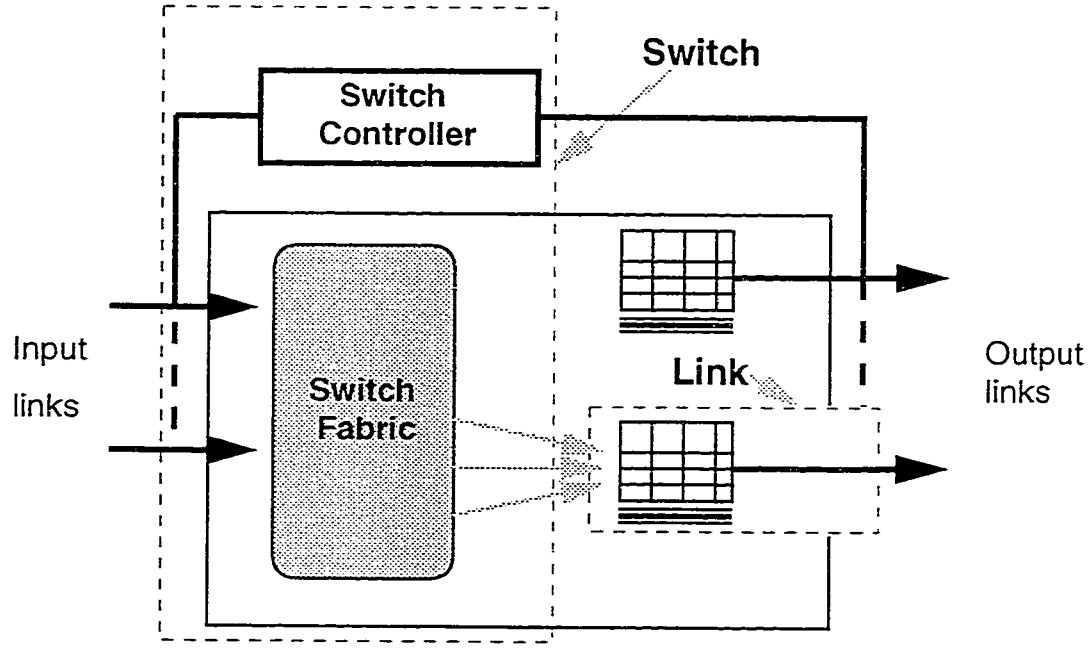


Figure 3.9 Model for an ATM Switch Node

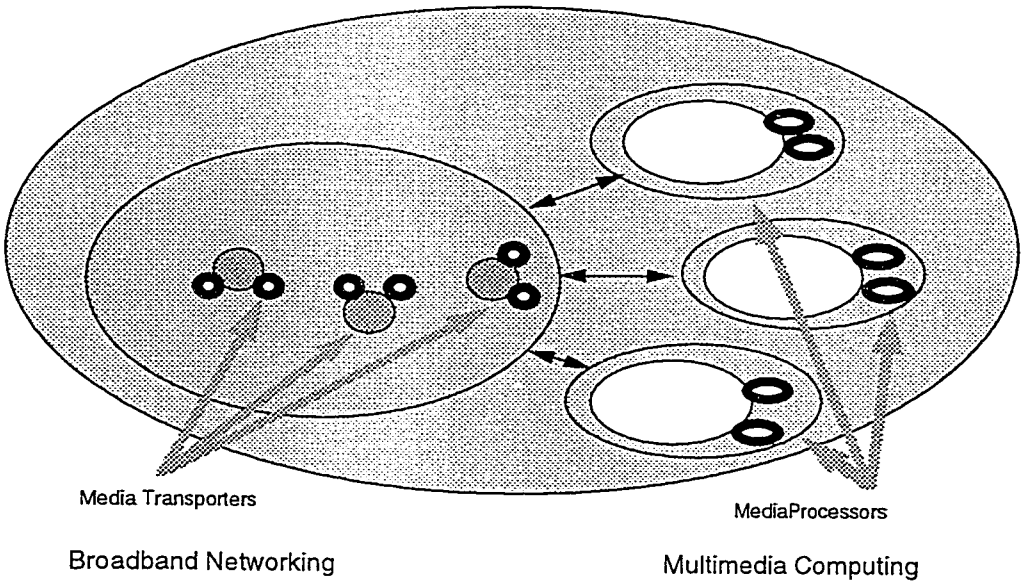


Figure 3.10 Multimedia Network Resources

Another basic interface in the BIB is the VirtualCapacityRegion. This interface characterizes the capability of multimedia networking resources with QoS guarantees. Every VirtualResource is associated with a VirtualCapacityRegion. The capacity of

networking as well as multimedia resources is characterized through the concept of schedulable and multimedia capacity region [LAZ95c], which gives a good method for modeling the quality of service abstraction of the broadband network. For more details, please refer to [HYM91] [HYM93].

3.3.4 Broadband Kernel Service

The services that are provided by the multimedia network for high-level multimedia service creation are referred to as broadband kernel services [LAZ96a]. These mainly include services for connection management, routing, device management and transport. Generally, these services are simply performed by interconnecting or binding the objects in the BIB. The connection management service provides connection setup and release between a number of network endpoints. The routing service provides route selection capabilities between two network endpoints. The connection service, together with the routing service, forms the basic connectivity services of the transport network. On the host end, the device management service is used for tracking and managing the various multimedia devices. The transport service is used for managing the transport streams in the network, mainly includes rate control and bandwidth renegotiation. Based on these broadband kernel services, let's see how the high-level multimedia services can be easily constructed in the Xbind system.

3.3.4.1 The Xbind Video Conferencing Service: An Example

The whole architecture of the Xbind video conferencing service has four layers (see Figure 3.11).

At the bottom layer is the BIB, a collection of interfaces that give the abstraction of the networking and multimedia resources of the binding architecture. The objects in the BIB correspond to the hardware equipments, such as ATM switches and multimedia

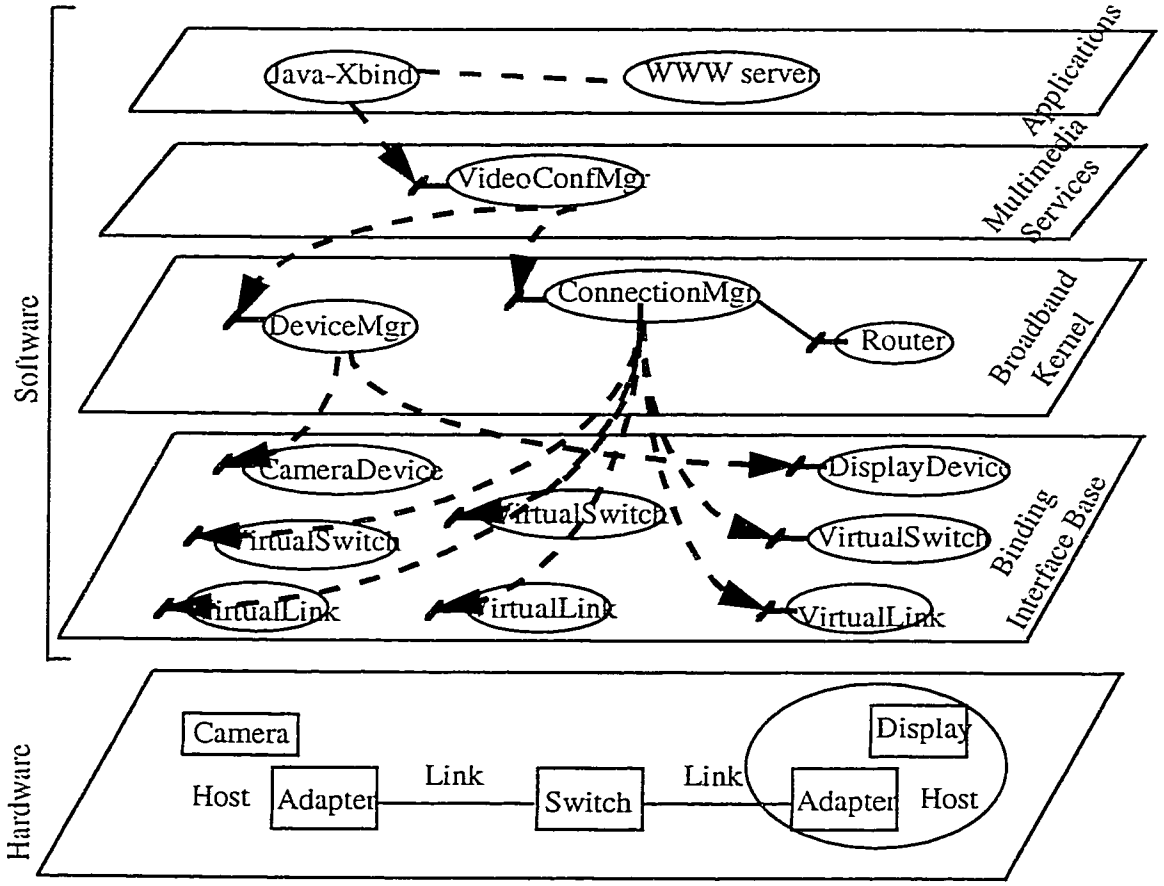


Figure 3.11 Java-Xbind

devices. Upon this layer is the broadband kernel service layer, in which services are used to build higher layer multimedia services. The multimedia services layer consists of very high level service, such as content search, retrieval, distribution and interaction. The video conference management service is one of the example. At the highest layer of the architecture is the user level applications, such as WWW browsers or other interactive applications.

According to the architecture, from the top most layer, when a client application requests a video connection, the call is translated from the Java applet into a CORBA call. In this case, the call is directed to the video conferencing service entity which in turn makes two calls to the broadband kernel. The first call is to the connection manager

whose role is to set up an ATM connection between the requested hosts. The second call is directed to the device manager to activate the multimedia devices at the end hosts. The connection manager in turn invokes the network resources such as the VirtualSwitches and the VirtualLinks to reserve the required bandwidth and buffer space to fulfill the requirements of the connection. After that, the device manager invokes the appropriate calls to the multimedia devices to initiate the generation and transmission of audio/video streams through the appropriate VPIs and VCIs, which are previously reserved by the connection manager. Reaching this point, the Java-applet is notified of the success or failure of its request.

Chapter 4

Signaling in Communication Networks

4.1 Evolution of Network Signaling

Signaling in communication networks is mainly concerned with the reservation and allocation of network resources and the provision of communication paths between end systems [WAD96]. To a large extent, signaling plays an important role in enabling the efficient use of network resources as well as the services that the network provides to its users. In a more formal manner, signaling can be defined as "the collection of procedures used to dynamically establish, maintain, and terminate connections, which require

information exchange between the network users and the switching nodes, and between switching nodes" [EDW96].

Generally speaking, network signaling provides the ability to establish, maintain and release connections between users and nodes within the network, based on which the efficient use of network resources and services can be achieved. Therefore, signaling has always been considered as an important aspect of the communication networks. The past few decades have seen the dramatic development of communication networks, which inevitably posed great effects and new requirements on the signaling systems. As networks become more and more complex, the functions performed by the network signaling necessarily grow. Consequently, signaling systems have also evolved a lot over the years to keep pace with the advances in the communication networks and their services [MIN89].

Commonly signaling can be classified into user-network, network-node, and user-user signaling, as shown in Figure 4.1. User-network signaling provides users with some control messages to pass to the network in order to establish and control calls with other users. Network-node signaling is used between network nodes to convey information for the establish and control of a connection, including the allocation and management of network resources. User-user signaling is used to convey control signals between end users [MIN89] [WU95].

In the traditional telephone networks, the signaling system is grouped into three categories of signals: supervisory, address and call information. The supervisor signals include the "on-hook/off hook" signal used to request service, answer an incoming call, or disconnect a call. Address signals carry the telephone number of the called party. The call information signals are audible tones and announcements, such as dial tone and busy signals, that convey progress in completing a call. In the early "ringdown" systems, with

these three category signals, the end-to-end connection is done through operator controlled

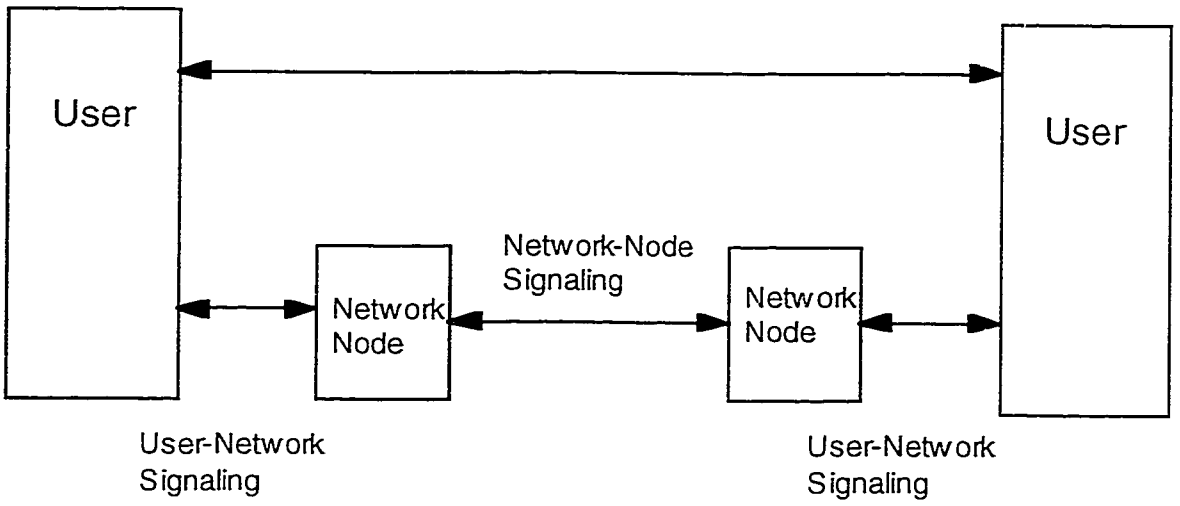


Figure 4.1 User-Network, Network-Node and User-User Signalling

switchboards. Later on, with the introduction of stored program control switch and deployment of computer networks, digital circuit switched networks emerged largely to support voice services. The current telephone networks are mainly constructed as circuit switched networks.

4.1.1 Signaling in Circuit Switched Networks

In the circuit switched networks, a dedicated communication path, called circuit, must be established between two end points before data transmission occurs. The circuit establishment and disconnection are performed by the signaling system. Early signaling approach in the circuit switched networks has been on a in-channel basis. With in-channel signaling, the same channel is used for signaling and voice transmission. There are two forms of in channel signaling: in band and out-of-band [STA95].

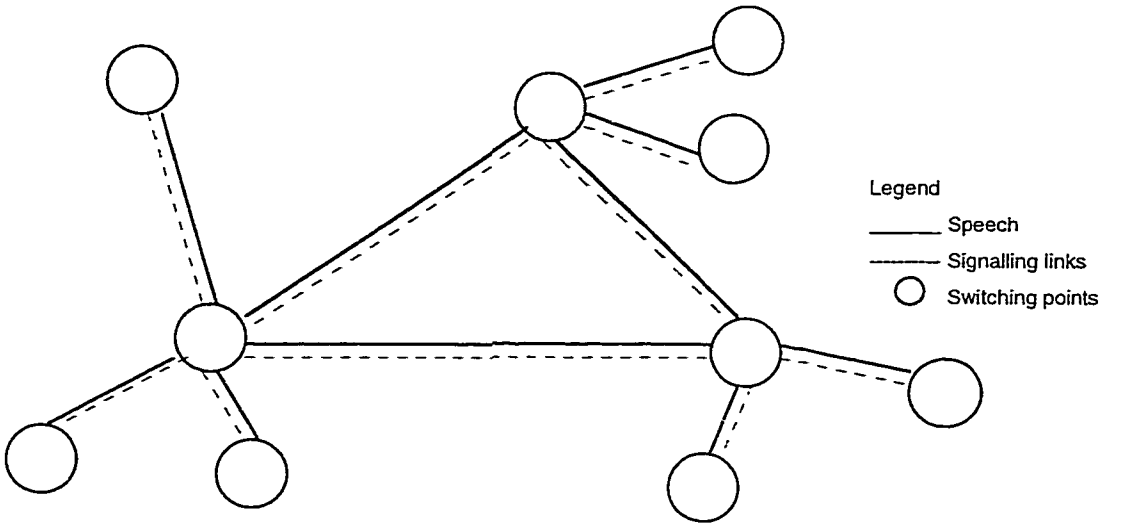
In-band signaling uses the same physical path and frequency band as the voice signals, while with the out-of-band signaling, a separate narrow signaling band is

allocated for control signals. In general, the transfer rates within these two approaches are very limited. With in-band signaling, the control signals can not be sent when there are voice signals on the channel. With out-of-band signaling, only a very narrow bandwidth available for the transmission of control messages. With such limits, in-channel signaling only works well for simple telephone networks and it is difficult to accommodate any but the simplest form of control messages. Therefore, to better support more services and to cope with the increasing complexity of evolving network technology, more powerful signaling approaches are on demand. Later on, the in-channel signaling is gradually replaced with common channel signaling approach.

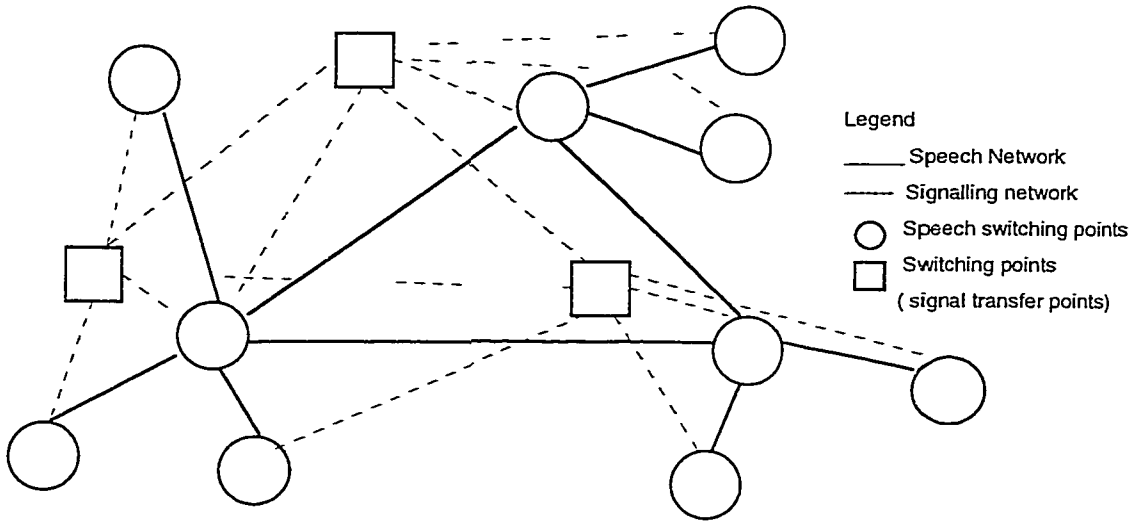
Common channel signaling is developed for modern telephone networks, in which control signals are carried over control paths being completely independent of the voice channels. There are two modes of operation used in common channel signaling: associated mode and nonassociated mode. In the associated mode, the common channel goes on the same trunk groups as all the channels under its control, while the nonassociated mode is more complex but more powerful. In the nonassociated mode, the control signals are transferred on a separate network which controls the switching nodes that serve the subscriber calls. In comparison with in-channel signaling, common channel signaling approach can greatly reduce the call set up time and it also more adaptable to evolving functional needs. Figure 4.2 shows the two modes.

4.1.2 Signaling in Packet Switched Networks

The key characteristic of circuit switched networks is that resources within the network are dedicated to a particular connection. This makes it more efficient for constant data rate traffic such as voice. However, for many computer based data communication applications, such as e-mail, connection-oriented service is not appropriate. To better support data communication, packet-switched networks emerged around 1970 [STA95].



(a) Associated



(b) Nonassocaited

Figure 4.2 Common Channel Signaling Modes

Packet-switched networks allow users to operate in either a connection-oriented mode, or in a connectionless mode. In a connectionless mode, there is no signaling involved for establishing and disconnecting connections. User packets, or called datagram, are routed to their destinations on a per-packet basis. A good example of such a network could be Internet, in which the Internet Protocol (IP) is used for routing packets [LAP94]. However, IP does not guarantee the reliable transmission of the user data. Packets may be lost or duplicated. To get reliable transmission, one can provide the

end-user with reliable transport services by building a reliable transport protocol on top of IP, such as TCP [COM91]. However, in connection oriented mode based networks, such as X.25, a route, called "virtual circuit", must be set up prior to data transfer, thus signaling system is needed for setting up and tearing down virtual circuits. Take the example of X.25, let's see how the signaling is performed in such a connection oriented packet switched network.

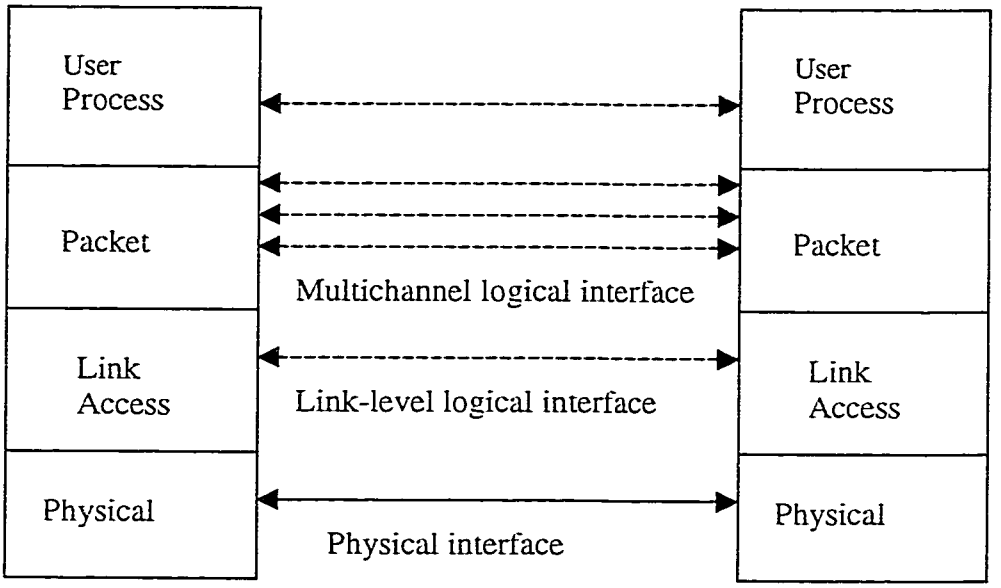


Figure 4.3 X.25 Interface

X.25 specifies three layers of functionality: physical layer, link layer and packet layer (see Figure 4.3). The packet layer is responsible for virtual circuit set up, tear down, and reliable end-to-end transmission. X.25 provides two types of virtual circuit: virtual call and permanent virtual circuit. A virtual call is a dynamically established virtual circuit using a call set up and call clearing procedure. A permanent virtual circuit is a fixed, network-assigned virtual circuit. A typical virtual call set up in X.25 can be illustrated below in Figure 4.4.

To set up a virtual call, the source DTE (Data Termination Equipment) first sends a call-request packet, as well as the required destination DTE network address, called

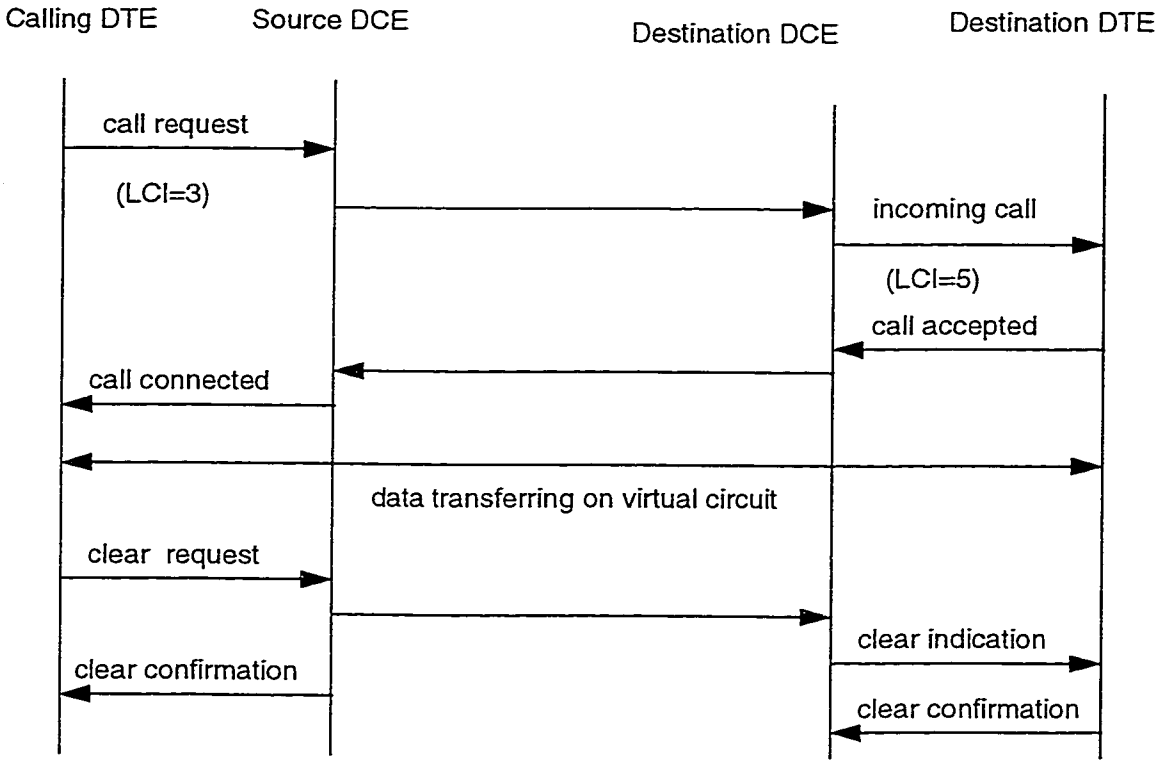


Figure 4.4. Virtual call set up and tear down of X.25 protocol

Logical Circuit Identifier (LCI), to the source DCE (Data circuit Termination Equipment), Then the request is forwarded to the destination DCE through the network. At the destination DCE, a second LCI is assigned to the call-request before it is forwarded to the destination DTE by an incoming call request. The destination DTE indicates acceptance of the call by sending a call-accepted packet to destination DCE. When the source DCE receives the message indicating the call has been accepted, it sends a call-connected message to the calling DTE. At this time, virtual circuit is established and the source DTE can start sending data packets.

4.1.3 Signaling in Integrated Services Digital Network (ISDN)

The extensive use of digital technology and deployment of circuit and packet switched networks have given impetus to a faster pace in the development of Integrated Services Digital Network (ISDN) [LAP94]. By providing a common interface to users, ISDN is

intended to integrate different networks, such as circuit switched network, packet switched network, LAN and WAN and to provide a wide variety of services, including voice, data and teleservices as well, such as telefax and televideo. Therefore, in order to support not only circuit switched connection but also packet switched connections, signalling plays an important role in the ISDN networks.

Similar to X.25, ISDN architecture specifies three functional layers: physical, data link and network layer, in which the network layer is responsible for establishing and releasing connections. In ISDN, an out-of-band signaling channel, called D-channel, is used and dedicated for signaling, while B channel is used for the transmission of data. Usually the connection is set up on B channels using the D channel signaling protocols, among which Q.931 and ISDN-UP (User Part) of CCSS7 (Common Channel Signaling System No. 7) have been standardized as user-network signaling and network-network signaling protocols, respectively. For in-depth description, please refer to [STA95].

Generally speaking, the emergence of ISDN has been considered as an evolution of communication technologies. However, lack of bandwidth available to users is considered as the major limitation of ISDN. Later on, with the high transmission capacity offered by optical fibers and the increasing demand for high-speed communications, a large effort was put into the development of Broadband Integrated Services Digital Network (B-ISDN), in which optical fibers are deployed as a primary choice of medium in providing the infrastructures for the network [KAW91]. Following ISDN, the target of B-ISDN is to merge the disparate set of networks that exist today, into a single, unified infrastructure capable of supporting all types of communication services. A new cell-based transport technology called Asynchronous Transfer Mode (ATM) is adopted for B-ISBN (see Chapter 2). Correspondingly, the emergence of broadband ISDN had a profound impact on signaling systems and architecture.

4.2 Signaling in Broadband Networks

The Broadband ISDN promises to offer customers a vast array of services by means of optic transmission, such as multiparty multimedia communications and interactive video distribution [TU92]. The objective to support the increasing diverse of services in turn poses great demand for much more powerful and flexible signaling systems for B-ISDN. According to [EDW96], signaling in broadband networks "should have flexibility and resilience and support the wide variation of both configuration and bandwidth that is possible in an ATM B-ISDN". In recognition of these, much standardization and research effort has been carried out.

4.2.1 Standard Broadband Signaling

To better support various services that are supposed to be offered by broadband ISDN, such as multimedia services, standardization efforts are mostly centered on the extension of existing narrowband signaling protocols. To allow for the different communication scenarios, the separation between call control and connection control is specified. That is, connections can be set up and released during a call. To meet the requirement of users for different calls, the signaling system of B-ISDN should have the following capabilities [LAP94] [KAW91]:

- Simultaneous establishment or release of multiple connections belonging to a call;
- Add and remove connections to and from an existing call;
- Add and remove a party to and from a multiparty call;
- Correlate connections comprising a multiconnection call;
- Reconfigure a multiparty call including an already existing call or splitting the original multiparty call into more calls.

In keeping with these capabilities, signaling for call and connection control is considered in the control plane of B-ISDN reference model (see Chapter2.3), in which two protocols are defined, i.e. Q.2931 at UNI for user-network signaling and B-ISUP at NNI for network-network signaling. They work together to set up on-demand Switched Virtual Circuit (SVC) connection across the ATM-based broadband networks.

Q.2931 is an extension of ISDN Q.931 signaling protocol. Different set of signaling messages are defined, including call establishment messages, call clearing messages and point-to-multipoint messages [WU95] [STA95]. With these messages, users can also specify the required QoS parameters, such as bandwidth, cell loss, and so on. A possible point-to-multipoint call set up sequence is given in Figure 4.5.

The root user starts to set up a point-to-multipoint connection by sending a SET-UP message. When accepted, the network node sends a CALL_PROCEEDING message back, and the SET_UP message is forwarded to the called party. The called party accepts the call by sending a CONNECT message, which is forwarded back to the root. After the point-to-point connection has been set up, the root can issue multiple ADD_PARTY requests to add all the participants. An ADD_PARTY request may be rejected by the network due to insufficient resources, or rejected by the called party by an ADD_PARTY_REJECT message. When accepted, the called party sends an ADD_PARTY_ACK message, which is forwarded to the root. When the connection is set up, the root can start sending data packets.

At the NNI interface, B-ISUP protocol is defined, which is an extension based on the ISDN User Part (ISDN-UP) of the Signaling System No. 7 (SS7). The connection establishment is made on a hop by hop basis through the use of B-ISUP protocol and the associated call processing software in broadband switching systems.

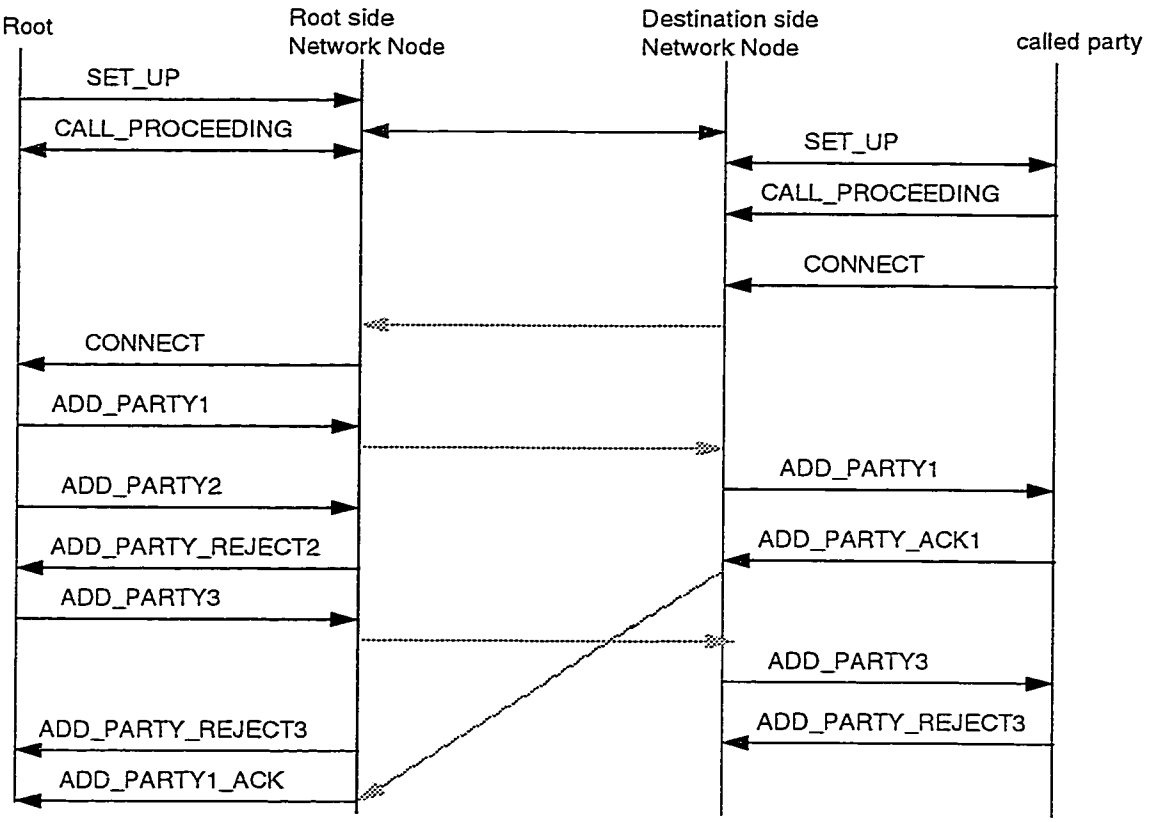


Figure 4.5 Point-to-multipoint call setup in Q.2931

Examining these two protocols, we may see some limitations of existing standard signaling approach. First, the call/connection control services that are supported by it are only limited to the basic point-to-point and point-to-multipoint ones. However, many new applications may require more services other than point-to-point and point-to-multipoint. For example, a conference call may need many-to-many communication scenario, and a distributed data collection system may need many-to-one services [BUB91]. Second, the hop-by-hop based connection establishment along the network nodes through the B-ISUP protocol does not lead to easy extension for multiple connections and the end-to-end connection set up delay is great [VEE95]. Therefore, to support more applications and make the signaling systems more efficient, much research effort has been made.

4.2.2 New Development of Broadband Signaling

The research in the broadband network signaling field can be summarized into the following aspects: how to model the signaling system; how to improve the efficiency of a signaling system; and how to better support the multimedia multiparty services.

As far as how to model the signaling system is concerned, "An Efficient Signaling Structure for ATM Networks" is proposed by [WU96]. And also with the adoption of object-oriented paradigm, "An Object-Oriented Resource Model for Supporting Signaling and Control of Broadband Services" is proposed by [TU92].

In terms of improving the efficiency of the signaling system, to overcome the inefficiency which is caused by hop-by-hop connection establishment paradigm, a "client-server" based signaling approach is proposed by [VEE95], which allows for the easy handling of multiparty connection and improves end-to-end connection set up delay by parallelizing certain operations that occur during connection establishment.

As to the better support for multimedia services, notable works are the "Connection Management Access Protocol (CMAP)" [BUB91] from the Applied Research Laboratory at Washington University, and "advanced multimedia, multipaty, multiconnection services" [MAG95] [MAG94] [MAG93] from MAGIC RACE project, Europe. In both approaches, some extensions have been made to the current Q.2931 and B-ISUP protocols, and the separation between call control and connection control has been greatly enhanced.

4.2.2.1 CMAP

The Connection Management Access Protocol (CMAP) [BUB91] is targeted for managing multipoint connections in high-speed packet switched networks, aiming at the

provision of not only point-to-point, point-to-multipoint call services but also many-to-many and many-to-one services to meet the requirement for more new applications.

CMAP can be viewed as an extension of Q.2931. Based on a call model proposed for representing multipoint connections in a broadband network, CMAP is used for network clients' creating, manipulating, and deleting calls. Within the call model, a call is defined as a set of one or more communication channels between two or more clients. A multipoint call is a call involving two or more clients; a point-to-point is a special case of a multipoint call involving only two clients.

When a call is created, a single connection is established between the network and the client who creates the call. This client is designated the owner of the call. Additional clients can be joined by: 1) Innovation from the owner, where the invited party has the option of refusing the innovation. 2) Request from a client where the owner has the option of denying the request, or 3) Request from a third party, who is not necessarily in the call, to add a client. Once a call has been created, additional connections and clients can be added to, or deleted from, the call.

Without going into the details, as we can see that by manipulating the call and connections within the call, users can flexibly construct communication channels for their special needs.

In addition to the CMAP protocol at the user-network interface side, the Connection Management Internal Protocol (CMIP) is also used by nodes within the network to manipulate connections at the NNI interface. CMIP is used to flexibly support a wide range of bandwidth, QoS and efficiently manage network resources.

4.2.2.2 RACE MAGIC

The multi-aspect (multi-party and multi-connection) will be the inherent part of future B-ISDN Services. This adds an extra level of complexity to the control architectures. To keep with this, the objective of MAGIC project from RACE is to develop signaling protocols for the future B-ISDN, capable of handling multimedia, multiparty, multiconnections services [MAG93] [MAG94] [MAG95].

It has been widely agreed that future B-ISDN should support Call and Connection separation. As a result of it, three control protocols have been proposed. The call control (CC) protocol should allow end-users to negotiate the communication configuration they desire. The Connection control, also called Bearer Control (BC) protocol, is responsible for manipulating the switching and transmission resources that are required for implementing the communication configuration. Additionally, considering the complexity introduced by multimedia, the need for an additional protocol to control the newly introduced network resources, such as bridges and converters, is also specified. This results in the introduction of a new protocol called Resource Control (RC). The overview of the MAGIC signaling architecture can be illustrated as in Figure 4.6.

1. At the User-Network Interface between the user and the network node, a Call Control Interface is provided. Based on Q.2931, the Call Control interface extends it to encompass the multi-party, multimedia requirements of the broadband network.
2. At the Network Node Interface between different network nodes, a Call Control (CC) Protocol is provided for the edge to edge call control.
3. At the Network Node Interface between the specialized resources, such as bridges, a Resource Control (RC) protocol is provided for the network control of specialized resources.

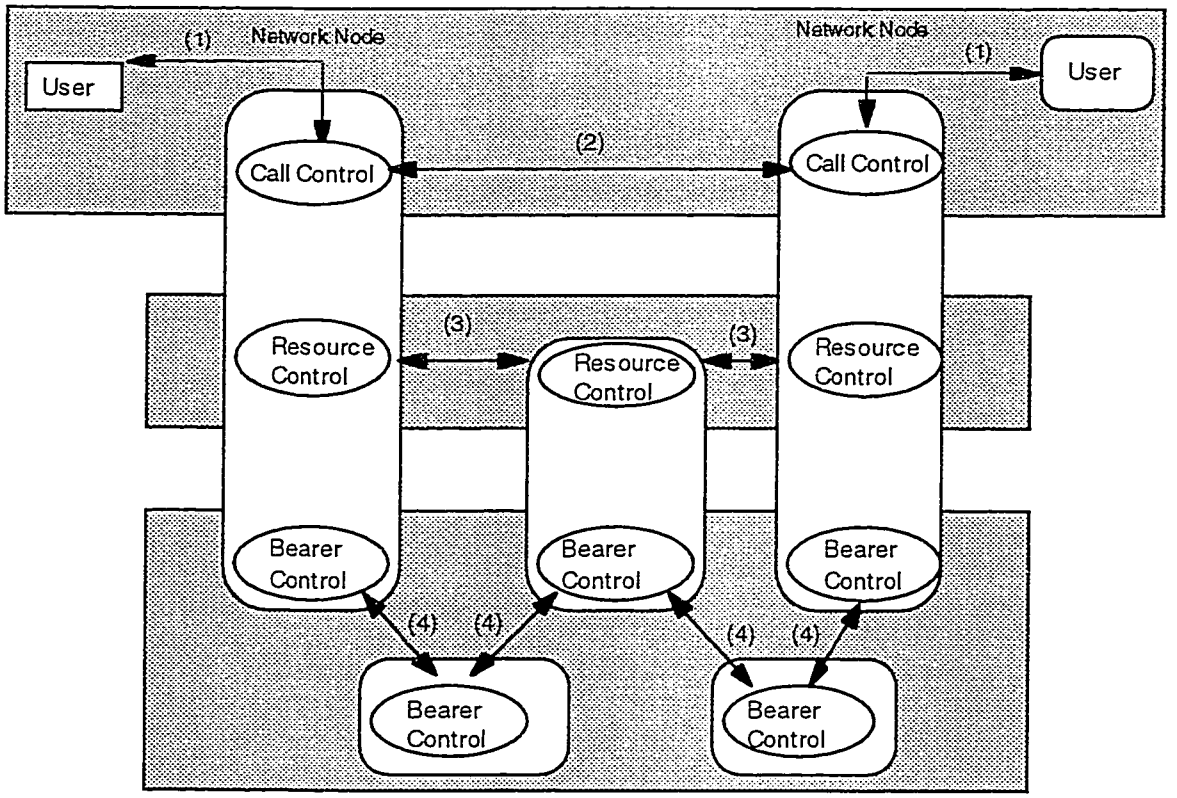


Figure 4.6 MAGIC signaling architecture

4. At the Network Node Interface, a Bearer Control (BC) protocol is provided for the link by link control of connection, which is based on the B-ISUP protocol and also some extensions are made.

In a whole, the on-demand multimedia and multiparty services are expected to become the driving force behind B-ISDN. This poses a formidable challenge for its underlying signaling system, which should not only provide users with an efficient interface to set up and dynamically modify complex multimedia, multiparty, multiconnection calls, but also provide the means for the network to make connections that fulfill the wishes of the users. From this point of view, CMAP and RACE MAGIC provide some good attempts. However, they are still limited to the traditional signaling protocol paradigm, which is based on the traditional assumption of low-level intelligent Customer Premises Equipment (CPE) and the UNI/NNI model. The signaling

capabilities are commonly provided by the network providers in a proprietary manner. In recent years, rapid service delivery has become a main challenge facing the telecommunication society, while the signaling service can be considered as the basis for the creation and deployment of advanced services. Under the UNI/NNI model, the involvement of signaling services in any third-party service programming is only restricted to customizing a set of operational parameters to build up signalling messages. From this perspective, the traditional signaling stack approach is not suitable for the easier service creation. in order to allow for the rapid creation and deployment of new services, the signalling system of ATM based broadband network needs to be designed for flexibility and scalability. This, along with the new development of Distributed Object Computing technology, leads to a major shift of "open network control" based "open signalling" approach in broadband networks.

4.3 Open Signaling for Broadband Networks

With the objective of being more flexible and scalable for service creation and deployment, the "open signalling" approach is proposed to exploit the Connection Management service as a high-level programming entity based on the deployment of Distributed Object Computing technologies, such as CORBA. More specifically, in this approach, the Connection Management service is implemented as a high-level CORBA object and specified in CORBA IDL. The network entities are modeled as CORBA objects as well.

Since the Distributed Object Computing technology, such as CORBA, makes it possible for the hardware entities to be programmed as high-level objects with well-defined interfaces and the capabilities for controlling, accessing as well as managing them are given by the interfaces, the "open signaling" approach implies a separation between the hardware switching functions and the call/connection functions in broadband

networks. In other words, the implementation of network control functions should be independent of network entities, which provide the basic communication functionalities.

The key factor of the "open signalling" approach lies in the "open network control" interfaces available from the network providers. Based on the low-level generic control interfaces provided by the network providers, it is possible for third party service providers to develop their own control systems in a hardware independent manner and the connection setup and removal can be expressed in terms of interconnecting (or binding) high-level signaling entities (objects), which model the network entities and run on a general purpose distributed computing platform, such as CORBA, as we can see from the TINA architecture and the Xbind system. In the TINA architecture, with the separation of the services and connection control in its service architecture, the introduction of new services is greatly facilitated independently of the underlying network. In Xbind, it allows the installation of signaling software on an ATM switch from a third party vendor, thus simplifying the service creation, deployment and management. In the sense, "open signalling" approach is also termed as a "hardware independent signalling" approach.

In summary, the "open signalling" approach prevails over the traditional signalling protocol stack in the following aspects [DEV97] [LAZ97]:

Firstly, "open signalling" exceeds the limitations of low-level UNI/NNI model, on which the traditional signalling protocol stack is based. As mentioned before, the inflexibility and rigidity of the UNI/NNI model can be seen in terms of utilization of network services, such as connection management, for high-level service creation, which is only limited to customizing only a small set of operational parameters for composing signaling messages. However, with the support of distributed object computing platform, "open signalling" approach makes it possible for the "signalling" operation to be

constructed as a high-level computational object. So that the service creation can be done within a high-level programing paradigm. This makes "open signalling" approach much more flexible and scalable.

Secondly, the "open signaling" approach devolves the management and control of switches and end devices into a coherent external system. This enables customers to configure their systems in a manner best suited to their needs, so that the more efficient utilization of network resources can be achieved.

Thirdly, the "open signaling" approach enables new management and control policies to be introduced in the network without modifying switch equipments, and this will allow legacy and new signalling and management systems to co-exist.

In reality, the concept of "open signaling" was proposed by the COMET group at Columbia University with the launch of Xbind project in 1994. Being targeted for a "broadband kernel", the emphasis of Xbind "open signaling" is put on the control of ATM switching nodes. A set of low-level interfaces, called "open APIs for ATM switch control" are proposed. The APIs are divided into three distinct categories. The Switch Fabric APIs define interfaces for the manipulation of the switching tables, allowing for the control of connections. The QoS Control APIs provide the ability to specify QoS in terms of predefined traffic classes. The switch Configuration and Management APIs is used for switch management, such as resetting of hardware components, and functions for switch management, such as the checking of a component status.

Since the "open signaling" approach was first proposed by Columbia University in 1994, a number of new projects and international forums, for example, OPENSIG and OPENARCH, have been launched to promote the ideas. The work from Lancaster University also extends the signaling into the end system. It is envisioned that end-system signaling must not only consider the provision a suitable interface for its control, but

must also involve understanding the requirements of "local" signaling used for the co-ordination and control of resources in the end system. The DCAN project (Devolved Control ATM Networks) from Cambridge University is intended to devolve the management and control of switches and end systems into a coherent external system. It is believed that the control and management functions of the various devices that make up such a network- typically ATM switches- should be extracted from the devices themselves and be delegated to external workstations dedicated to that purpose [DEV97], so that the customers can configure their systems in a manner best suited to their needs. Also by introducing an "open network control interface", a Service Execution Environment (SEE) from Helsinki University of Technology [RAS97] is developed for easier service creation.

Chapter 5

An Open Signaling System over VIVID ATM Switches: Implementation

5.1 Introduction

This chapter presents the implementation of an "open signaling system over VIVID ATM switches". Along with the development of the Xbind project, the COMET group at Columbia University first gave an implementation of an "open signaling" system using FORE Systems switching technology around 1995. In the early products of ATM switches, the operating systems are UNIX based. As we mentioned before, UNIX is only designed to offer fair use of system resources among competing processes, but it is not

suitable for developing time-critical applications. However, to meet the high-performance and high-speed requirements of ATM switches, the software systems of them need to be designed as real-time applications. In recent years, with the availability of high-performance real-time operating systems, such as VxWorks, pSOS, and Chorus, most of the ATM switch vendors have been gradually migrating their products to the real-time based operating system. Due to the advantage of real-time OS features, such as multitask scheduling and intertask communications, it is envisaged that in the future most of the ATM switches in the market, although from different vendors, will be based on real-time OS. In this case, deployment of such an "open signaling" system over a real-time environment is of much more significance. With the gradual shift to the new software architecture for easier service creation and deployment in the telecommunication society, it is believed that our work will be a good attempt for scaling such new software architectures into the real-time environment.

In the following, we give the description of the implementation of an "open signaling system over VIVID ATM switches" using CORBA. Since the initial objective of this project is to port Xbind to the real-time OS based multi-vendor ATM networking environment, some of our work is based on the Xbind system.

5.2 System Architecture

The whole system can be described as in Figure 5.1.

On the platform side, as in typical ATM networks, workstations equipped with ATM adapter cards are connected via VIVID ATM switches. The operating system for workstations is UNIX based Solaris 2.x from SUN Microsystems Inc. The operating system on the VIVID ATM switches is real-time based VxWorks 5.3 from Wind River System [WIN96]. To deploy such a system over a CORBA platform, each workstation and VIVID ATM switch has a CORBA/Orbix daemon running. Due to the difference

between the UNIX and VxWorks operating systems, we have Orbix for UNIX 2.0 and Orbix for VxWorks 1.3.5 from IONA Technologies Ltd. installed on workstations and on VIVID ATM switches, respectively.

On the software side, the system is designed as a distributed "client-server" architecture, consisting of Connection Manager and NodeServers. The Connection Manager functioning as the "client" can run on any workstations. On each VIVID ATM switch, there is a NodeServer, which intercepts and deals with the CORBA requests from the Connection Manager. The interactions between the Connection Manager and NodeServers are enabled by the CORBA/Orbix platform. Since the software is written in C++, the language mapping from CORBA IDL to C++ is chosen.

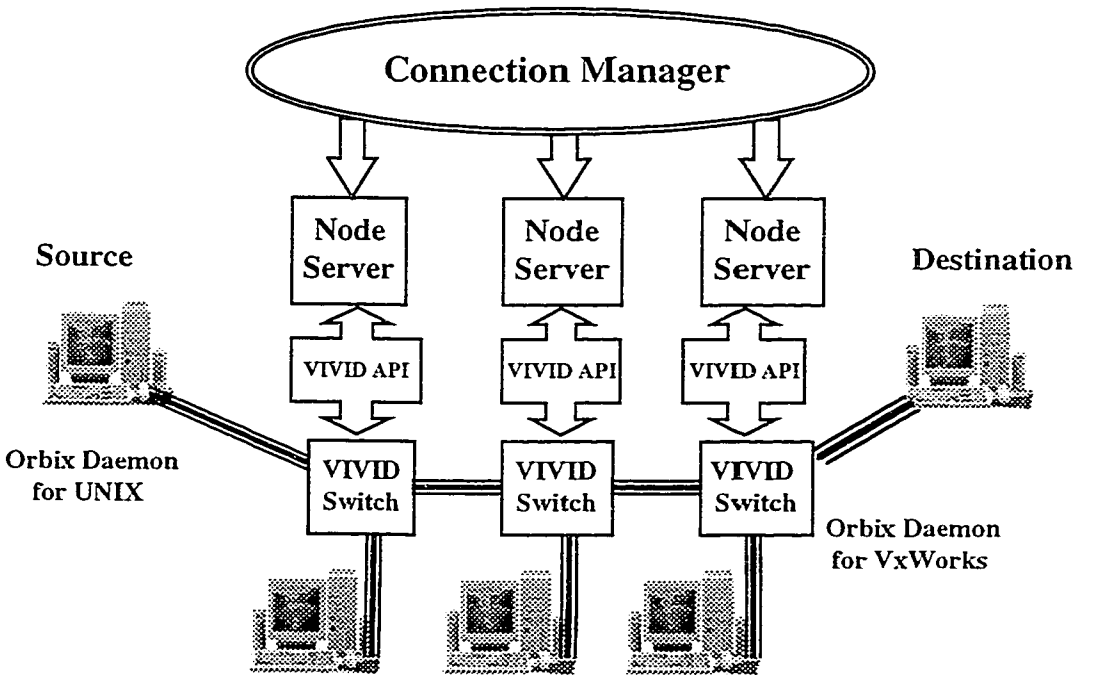


Figure 5.1 An Open Signaling system over VIVID ATM Switches

The interaction pattern between the connection manager and the NodeServers is that the Connection Manager communicates directly with all the NodeServers and distributively controls them to set up virtual channels in the corresponding switch node.

Our work for the implementation of such a system can be divided into the following aspects and stages:

- deployment of CORBA/Orbix over a real-time OS environment
- development of open APIs for VIVID ATM switch control
- development of NodeServer
- modifications of the Xbind Connection Manager

5.2.1 Deployment of CORBA/Orbix Over Real-Time Environment

Because such an "open signaling" system is CORBA platform based, the first step of our work is deploying and testing a CORBA/Orbix environment over the VxWorks based VIVID ATM switches. To put our description in proper context, we give a simple introduction of the VIVID ATM switch and VxWorks operating system as well.

5.2.1.1 VIVID ATM Switch -- CS3000

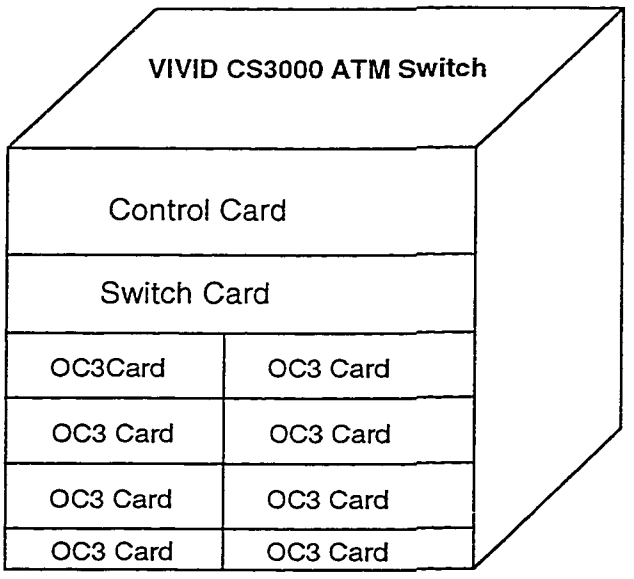


Figure 5.2 VIVID CS 3000 ATM Switch

The VIVID CS3000 [VIV97] is a powerful 6.4Gb/s ATM LAN switch that supports PVC and SVC connectivity. It can be fully configured with two Control cards, two Switch cards and up to eight quad OC3 interface cards (see Figure 5.2).

Each OC3 card has four ports on it, which are organized into one "slot". It is the port that provides the physical interface connecting other equipmenst, such as workstations and PCs which are equipped with ATM adapter cards, or other ATM switches. Maximally, the CS 3000 switch can support 32 ports. The VPI/VCI ranges for each OC3 port are shown in Table 5.1.

Connection Type	VPI	VCI
VCC	0	1 to 1023
VPC	1 to 15	0 to 450

Table 5.1 VPI/VCI Ranges for OC3 Ports

Certain VPI/VCI combinations are reserved for system use, for example, 0/5 is used for SVC signaling. Each OC3 port is identified by the slot into which they are inserted. For example, the first port on the first OC3 slot is identified by 1-1. Table 5.2 lists part of the port identifiers.

In addition to the 32 OC3 physical ports, there are two ports on the Control Card: the local management port and the Ethernet port. The local management port is used for the connection of a VT100 terminal to the switch to configure and manage it. For this purpose, a management session called "NMTI" (Network Management Terminal Interface) is provided.

Within each switch node, virtual connections are set up or torn down between endpoints. Each endpoint is identified by a OC3 slot and a port number, followed by a semicolon and the VPI/VCI identifier. i.e. slot-port; vpi/vci.

Slot Type	Slot identifier	Port Identifier
Control	CTL	SER_PORT_1
		ETHERNET
OC3	1	1-1
		1-2
		1-3
		1-4
OC3	2	2-1
		2-2
		2-3
		2-4
...	...	...
OC3	8	8-1
		8-2
		8-3
		8-4

Table 5.2 Identifiers for Ports

5.2.1.2 VxWorks: A Real-time Partner for UNIX

VxWorks [WIN93] is a high-performance real-time operating system and also provides a powerful development environment for real-time applications. VxWorks is not designed as a single operating system that "does it all", but rather, it is designed as a real-time partner for UNIX. As we know, the UNIX operating system has proven itself to be an excellent system for program development and for many interactive applications. But, it is poor at supporting real-time applications. The VxWorks operating system developed by Wind River is intended to cooperate and combine with UNIX and let each do what it does best. That is, VxWorks handles the critical real-time chores, while UNIX is used for program development and non-time-critical applications.

Usually, software development for a real-time application begins on the UNIX development system. As a cross-development host, UNIX is used to edit, compile, link, and store real-time code, which is then run and debugged on VxWorks. The applications themselves can make use of the many libraries supplied by VxWorks. After the real-time applications are loaded onto the VxWorks target board, the resulting VxWorks-based applications can then be run stand-alone with no further need for the host system. VxWorks and UNIX work well together because VxWorks is designed to be UNIX compatible at many levels, especially in extensive networking facilities.

As in all modern real-time systems, VxWorks is also based on the complementary concepts of multitasking and intertask communications. A multitasking environment allows real-time applications to be constructed as set of independent tasks, each with its own thread of execution and set of system resources. The intertask communication facilities allow these tasks to be synchronized and to communicate in order to coordinate their activities. Specifically, under the VxWorks environment, any subroutine may be "spawned" as a separate task with its own context and stack. The VxWorks multitasking kernel uses interrupt-driven, priority-based task scheduling. It is characterized by fast context switch times and low interrupt latency. For intertask communications, VxWorks supplies several types of traditional mutual-exclusion mechanisms, i.e. semaphores. VxWorks semaphores are fast and efficient. In addition, a fast and flexible message queue facility, intertask pipes, sockets, and signals are also supplied.

5.2.1.3 Integrating Orbix for VxWorks in a VIVID ATM Switch System

To also allow for distributed object computing over a real-time environment, not only limited to a UNIX environment, IONA Technologies Ltd. provides a specific version of Orbix for VxWorks 1.3.5. In terms of integration of diverse hardware from different

manufacturers into a single network and common distributed computing platform, this effort is of most significance.

Despite the real-time feature of VxWorks which makes it different from UNIX, the absence of a file system in VxWorks can be regarded as another major difference of it from the file based UNIX. As a result, there is a number of differences between the version of Orbix for VxWorks and the version of Orbix for UNIX, which can be outlined below [ION97].

First, due to the lack of a file system in the VxWorks installation, two basic components of the CORBA architecture, i.e. Implementation Repository and Interface Repository, are not implemented in Orbix for VxWorks, since the implementations of them largely rely on the use of a local file system. Usually, from Orbix for UNIX, the server can be activated in several modes and the Implementation Repository is used to store information about a server's activation mode and access permission. In Orbix for VxWorks, the server must be "spawned" as a task manually, in the sense that there is only one activation mode available, i.e. persistent.

Second, due to the absence of a file system in VxWorks, such a file as "/etc/hosts" on common UNIX systems, which can be consulted to determine which IP address a given host name maps to, is not available in VxWorks. Therefore, in order to make a host name known to VxWorks, it is important to use "hostAdd" commands to register them with VxWorks under VxWorks shell. For example,

```
-> hostAdd "switch.genie.uottawa.ca", "xxx.xxx.xxx.xxx"
```

Third, normally, in Orbix for UNIX version, Orbix is configured through the use of Orbix.cfg file to set environment variables. Because there is no file system support in VxWorks, the configuration of Orbix for VxWorks has to be done via the use of "putenv" command in the VxWorks shell before the orbix daemon is started. Such as :

```
-> putenv "IT_CONNECT_ATTEPTS = 5"
-> putenv "IT_DAEMON_PORT = 1570"
-> putenv "IT_DAEMON_SERVER_BASE = 1600"
-> putenv "IT_DAEMON_SERVER_RANGE = 100"
-> putenv "IT_LOCAL_DOMAIN = genie.uottawa.ca"
```

What deserves to mention here is that, in reality, when we tried to install Orbix for VxWorks 1.3.5 in the VIVID ATM Switch in the early stage of our work, we found that there are some incompatibilities between the system requirement of Orbix for VxWorks from IONA and the environment that we had for the VIVID ATM switch. As a result, our work is not just to install the Orbix for VxWorks according to the installation manual that we got from IONA, but rather, how to integrate the Orbix for VxWorks object code "ppc604.o" and "ppc604e.o" into the software system of VIVID ATM switches. That was the biggest challenge that we met. It was really a tough period but a very good experience. After almost two months of hard work and with the great help from the people in the VIVID group, we finally succeeded getting the daemon running on the VIVID ATM switch. For proprietary reasons, we won't give a detailed description here.

```
================================================================

[ Orbix daemon task initialization.....
[ -1.3.5: Orbix Version v1.3.5 for GNU C++ 2.7.2 on VxWorks 5.3
PowerPC 604]

Daemon Configuration:]
[ Daemon Port                          :1570 ]
[ Daemon Port Base                     :1600 ]
[ Daemon Port Range                    :100 ]
[ Local Host                           :switch.genie.uottawa.ca ]
[ Local Domain                         :genie.uottawa.ca ]
[ Message Queue Length                 :20 ]
[ Orbix Errors Module                  :LOADED ]
[ Orbix Internal Caching               :ON ]
[ Local communications optimisation :ON ]

[orbixd: Server "IT_daemon" is now available to the network]
[     Configuration Orbix-TCP/1570/Orbix-XDR ]

================================================================
```

When the "orbixd" daemon is started using "taskSpawn "orbixd", 100, 0, 1500, _orbixd" under VxWorks shell, we will see the above message on the screen:

The message, "Orbixd: server "IT_daemon" is now available to the network", indicates that the orbix for VxWorks daemon is successfully running on the switch, and ready to accept CORBA messages sent from other orbix daemons along the network. After we finished testing such an environment, we focused on developing the open APIs for the VIVID ATM switch control. This is the fundamental part of our work and many technical supports were given by the VIVID group.

5.2.2 Implementation of Open APIs for VIVID ATM Switches Control

Referring to the document entitled "Towards an Open APIs for ATM Switch Control" [LAZ96a] drafted by the COMET Group at Columbia University, and also taking into account the uniqueness of the VIVID ATM switch system, we gave implementations of the following two categories of APIs: Switch Fabric Control APIs and Switch Configuration and Management APIs. Generally speaking, the switch fabric APIs define those low-level interfaces for the manipulation of the switching tables, allowing for the control of point-to-point and point-to-multipoint connections within one ATM switch node, including addVC(), addP2MPVC(), removeVC() and removeP2MPVC(). The switch configuration and management APIs, to a lesser degree, define the basic functions for checking the hardware status, such as isPortOK(), queryPorts () and resetting the hardware components, such as resetPort (). In addition, we also developed some APIs to meet specific demands, such as getSwNodeIPAddress(), getNextVcEp(). These APIs provide the building blocks to be used to create higher-level connection management service.

5.2.2.1 Switch Fabric APIs

The switch fabric APIs play a key role in enabling the open control of the ATM switches. These APIs provide fundamental interfaces that can be used to set up virtual channels in each ATM switch node, including point-to-point and point-to-multipoint. In this aspect, only point-to-point APIs, such as addVC() and removeVC(), are considered in the Xbind system. In our work, point-to-multipoint switch fabric control APIs are also developed.

In general, the connection setup and removal within a switch node is fairly complicated. As far as the virtual channel setup is concerned, to set up a connection between the Ingress endpoint and Egress endpoint, which are identified by the port number, VCI and VPIs, first, the resources of the specified port and the whole switch fabric are checked to see if the user application's desired QoS parameters, such as peak rate and cell loss, can be satisfied; if it is, the Ingress and Egress endpoints will be added as an entry into the hardware-level reservation tables. Such that the virtual channel within the switch node is set up.

Our development of the Switch Fabric APIs, and also the Switch Management and Configuration APIs largely relies on the VIVID ATM switch software system. After getting familiar with their software system, we found that the VIVID ATM switch software system provides us with a flexible low-level interfaces for the development of these APIs. For confidential reasons, it is hard for us to give implementation details here. Only the description of these APIs is outlined below.

- **point-to-point APIs**

addVC

```
int addVC( Port inPort, VPI inVPI, VCI inVCI
  Port outPort, VPI outVPI, VCI outVCI, QOSClass connClass);
```

removeVC

```
int removeVC( Port inPort, VPI inVPI, VCI inVCI
    Port outPort, VPI outVPI, VCI outVCI, QOSClass connClass);
```

addVC() is used to create a point-to-point virtual channel between the Ingress endpoint and Egress endpoint within the switch node according to the specified VPI, VCI and Quality of Service class of traffic. Conversely, removeVC() is used to remove a point-to-point virtual channel between the Ingress endpoint and Egress endpoint within the switch node according to the specified VPI, VCI and Quality of Service class of traffic. Correspondingly, in terms of point-to-multipoint, we provide the following two APIs.

- **point-to-multipoint**

addP2MPVC

```
int addP2MPVC( Port  rootPort, VPI  rootVPI, VCI  rootVCI,
  Port leafPort, VPI leafVPI, VCI leafVCI, QOSClass connClass);
```

removeP2MPVC

```
int removeP2MPVC( Port rootPort, VPI inVPI, VCI rootVCI,
  Port leafPort, VPI leafVPI, VCI leafVCI, QOSClass connClass);
```

addP2MPVC( ) is used to create a point-to-multipoint connection between the Ingress endpoint, which is considered as a root of a connection, and the Egress endpoint, which is considered as the leaf endpoint, within the switch node according to the specified VPI, VCI and Quality of Service class of traffic. Conversely, removeP2MPVC () is used to remove a point-to-multipoint connection between the root endpoint and the leaf endpoint within the switch node according to the specified VPI, VCI and Quality of Service class of traffic.

Additionally, in accordance with the draft of "open APIs for ATM switch control", we also developed two APIs for query purpose, i.e. queryInVC() and queryOutVC().

queryInVC

```
int queryInVC (Port *inPort, VPI *inVPI, VCI *inVCI,
        Port outPort, VPI  outVPI, VCI outVCI);
```

queryOutVC

```
int queryOutVC (Port inPort, VPI inVPI, VCI inVCI,
        VCList *outVCList );
```

queryInVC() is used to trace a virtual channel backward. For a virtual circuit connection, when the given Egress endpoint is specified, the corresponding Ingress endpoint is returned. Specifically, if the given endpoint is part of a point-to-point connection, its mate endpoint will be returned; If the given endpoint is the root of a point-to-multipoint connection, its first leaf endpoint will be returned; If the given endpoint is the leaf of a point-to-multipoint connection, its root endpoint will be returned.

queryOutVC () is used to trace a virtual channel forward. When the specified Ingress endpoint is a part of a point-to-point connection. its mate endpoint will be returned; If the specified endpoint is a root, all the leaf endpoints will be returned. If the endpoint is a leaf endpoint of a point-to-multipoint connection, its root endpoint will be returned.

5.2.2.2 Switch Configuration and Management APIs

In order for the switch to be managed or configured by the third party software, it is required that the basic Switch Configuration and Management APIs also be provided. Here we developed a few fundamental ones as needed, which are:

resetswitch int resetSwitch ( );

resetPort int resetPort (Port port);

isPortOK int isPortOK (Port port);

queryPorts int queryPorts (short *supportedPorts, Port *activePortList);

resetSwitch () and resetPort () are used to perform a limited reset of the switch and the specified port, respectively. isPortOK () is used to check if the specified port on the switch is active and operational. queryPorts () is used to get the maximum number of ports supported by the switch and which of them are currently active.

5.3 Development of NodeServers

5.3.1 VirtualSwitch Interface

As we mentioned before, the basis for the "open signaling" approach lies in the generic interface available for controlling the network entities, such as ATM switches. Based on the above open APIs for VIVID ATM switch control, the open control of the VIVID ATM switch is given by the "VirtualSwitch" Interface and specified in CORBA IDL.

The VirtualSwitch interface models the functionality of the physical VIVID ATM switch node. The methods in the interface allow for the set up and tear down point-to-point or point-to-multipoint virtual circuit connections according to the QOS traffic class within the node. Our implementation of the VirtualSwitch is based on the Xbind system, but considering the specialty of the VIVID ATM switch software system, some changes are made and new methods are added, such as commitP2MPchannel(), removeP2MPChannel(), queryInVC() , and etc. In addition, as to the connection setup and removal, we also incorporate the QoS class as a parameters into the commitChannel and removeChannel methods, which is not considered in the Xbind system. In so doing, the users can specify the desired QoS traffic class, either ABR or UBR or CBR or VBR,

for the connection set up or removal. The detailed description of VirtualSwitch interface is given in Figure 5.3.

In general, the functionality of the VirtualSwitch can be divided into three parts: reservation of VCI/VPI pairs in the input and output ports of the switch, connection commitment at physical switch hardware level and later removal. The VCI/VPI pairs reservation is done through the use of a input reservation table and a output reservation table. The methods used for this purpose are setInputChannelIdentifier() and getOutputChannelIdentifier(). To free the VCI/VPI reservation, the unsetInputChannel() and unsetOutputChannel() methods are provided. To set up point-to-point connection and point-to-multipoint connection, we developed commitChannel() and commitP2MPChannel() methods in VirtualSwitch interface, respectively. At the same time when the channel is committed, some changes also happened to the reservation table, i.e. reservation free up. Correspondingly, to tear down the channel, two methods are provided: removeChannel() for point-to-point connection and removeP2MPChannel () for point-to-multipoint connection.

The implementation of the above four methods of commitChannel(), commitP2MPChannel(), removeChannel() and removeP2MPChannel() are built upon the switch fabric APIs, i.e. addVC(), removeVC(), addP2MPVC() and removeP2MPVC(). Take an example, the implementations of committing point-to-point channel method and point-to-multipoint channel method are described as in Figure 5.4 & 5.5:

```
interface VirtualSwitch {

  exception InsufficientReservedBlocks {};
  exception MaxSetExceeded {};
  exception MaxGetExceeded {};
  exception InvalidVCI {};
  exception InvalidVPI {};
  exception InvalidPortId {};
  exception NoFreeVCI {};
  exception NoFreeVPI {};

BIBStatus blockVCIVPI(in VCI vci, in VPI vpi, in PortId pid)
    raises (InsufficientReservedBlocks, InvalidPortId, InvalidVCI);
  BIBStatus unblockVCIVPI(in VCI vci, in VPI vpi, in PortId pid);

  BIBStatus setInputChannelIdentifier(in VCI invci, in VPI invpi,in PortId inpid)
    raises (MaxSetExceeded, InvalidVCI, InvalidVPI, InvalidPortId);

  BIBStatus getOutputChannelIdentifier(inout VCI outvci, inout VPI outvpi,
                                       in PortId outpid)
    raises (MaxGetExceeded, NoFreeVCI, NoFreeVPI, InvalidVPI, InvalidPortId);

  BIBStatus unsetInputChannel(in VCI invci, in VPI invpi, in PortId inpid)
    raises (InvalidVCI, InvalidVPI, InvalidPortId);

  BIBStatus unsetOutputChannel(in VCI outvci, in VPI outvpi, in PortId outpid)
    raises (VCINotAllocated);

  BIBStatus commitChannel(in VCI invci, in VPI invpi, in PortId inpid,
                          in VCI outvci, in VPI outvpi, in PortId outpid,
                          in QOSClass connQOS)
    raises (InvalidVCI, InvalidVPI, InvalidPortId);

  BIBStatus commitP2MPChannel(in VCI invci, in VPI invpi, in PortId inpid,
                              in VCI outvci, in VPI outvpi, in PortId outpid,
                              in QOSClass connQOS)
    raises (InvalidVCI, InvalidVPI, InvalidPortId);

  BIBStatus removeChannel(in VCI invci, in VPI invpi, in PortId inpid,
                          in VCI outvci, in VPI outvpi, in PortId outpid,
                          in QOSClass connQOS)
    raises (InvalidVCI, InvalidVPI, InvalidPortId);

  BIBStatus removeP2MPChannel(in VCI invci, in VPI invpi, in PortId inpid,
                              in VCI outvci, in VPI outvpi, in PortId outpid,
                              in QOSClass connQOS)
    raises (InvalidVCI, InvalidVPI, InvalidPortId);


  BIBStatus queryOutputIDs(in VCI invci, in VPI invpi, in PortId inpid,
                           out AddrIdentifierList outIds)
    raises (InvalidVCI, InvalidVPI, InvalidPortId);

  BIBStatus queryInputIDs(in VCI invci, in VPI invpi, in PortId inpid,
                          out VCI outvci, out VPI outvpi, out PortId outpid)
    raises (InvalidVCI, InvalidVPI, InvalidPortId);
```

Figure 5.3 VirtualSwitch CORBA IDL Interface

```
BIBStatus VirtualSwitch_i::
commitChannel (VCI invci, VPI invpi, PortId inpid, VCI outvci, VPI
outvpi, PortId outpid, QOSClass connQOS, CORBA_Environment &IT_env) {

if (freeReservation(invci, invpi, inpid, outvci, outvpi, outpid) ==0)
{
    if (addVC(inpid, invci, invpi, outpid, outvci, outvpi, connQOS)) {
        addUsed(outvci, outvpi, outpid);
        return SUCCESS;
     }
    else {
         printf("Switch: Commit point-to-point Channel failed!\n");
         return APPLICATION_FAILURE;
    }
}
 else {
      printf("FreeReservation in P2P failed!\n");
      return APPLICATION_FAILURE;
 }
}
```

Figure 5.4. Implementation of commitChannel method (point-to-point)

```
BIBStatus VirtualSwitch_i::
commitP2MPChannel (VCI invci, VPI invpi, PortId inpid, VCI outvci, VPI
outvpi, PortId outpid, QOSClass connQOS, CORBA_Environment &IT_env) {

if (freeReservation(invci, invpi, inpid, outvci, outvpi, outpid) ==0)
{
  if (addP2MPVC(inpid, invci, invpi, outpid, outvci, outvpi, connQOS))
     {
       addUsed(outvci, outvpi, outpid);
       return SUCCESS;
     }
  else
    {
       printf("Switch: Commit point-to-multipoint Channel failed!\n");
       return APPLICATION_FAILURE;
    }
}
else
{
      printf("FreeReservation in P2MP failed!\n");
      return APPLICATION_FAILURE;
}
}
```

Figure 5.5. Implementation of commitChannel method (point-to-multipoint)

5.3.2 NodeServer

Based on the corresponding C++ implementation of VirtualSwitch Interface, we wrapped it with a common CORBA server, named as "NodeServer". The NodeServer and Orbix daemon are spawned as two tasks running on top of VxWorks operating system within

the switch node. It, working together with Orbix daemon, accepts and performs the CORBA request from the Connection Manager. Part of the code of NodeServer can be described as follows (see Figure 5.6):

```
#include "node.h"
#include <stdio.h>

NodeServer ()
{
   ORB_init ();

   // Create Node
   node_i* node = new node_i();

   CORBA_Orbix.collocated (0);

   // Open communication with Orbix daemon
   TRY {
     CORBA_Orbix.impl_is_ready("NodeServer",
            CORBA_Orbix.INIFINITE_TIMEOUT, IT_X);
   }
   CATCHANY {
      printf("Error calling Orbix.impl_is_ready:\n");
      IT_X.printf();
      ORB_end();
      exit (-1);
   } ENDTRY

   ORB_end();
}
```

Figure 5.6 Implementation of NodeServer

When the NodeServer task is spawned, it in turn creates a C++ object, called "node". The "node" object is responsible for the creation and initiation of the "VirtualSwitch" object, i.e. the C++ implementation of the VirtualSwitch Interface

Usually, the virtual switch has to be configured (or initialized) by the "node" object. The items that needs to be configured include the IP address of the switch node and the blocking table of the switch. The blocking table contains the Port/VCI/VPI entries that have been allocated by the current switch node, so that they can not be reserved or later allocated for any new connection anymore. This is important in keeping the correct connection history information of the switch node.

To configure the virtual switch, in Xbind, a configure file stored in the Fore ATM switch is used. However, as we said before that in VxWorks based ATM switch, we do not have such a file system available. Correspondingly, we developed two low-level APIs for the same purpose: getSwNodeIpAddr() and getNxtVcEp(), through which we get information needed directly for the current switch node. The way we use here is much flexible. The API of getSwNodeIpAddr () returns the IP address of the switch node, while getNxtVcEp () is used to return the next connected endpoint. By loop it, we can get all the endpoints that has been used in connection within the whole switch node.

In the following, a dynamic scenario of how the connection is set up within the switch node when the CORBA message comes in from the Connection Manager is illustrated in Figure 5.7.

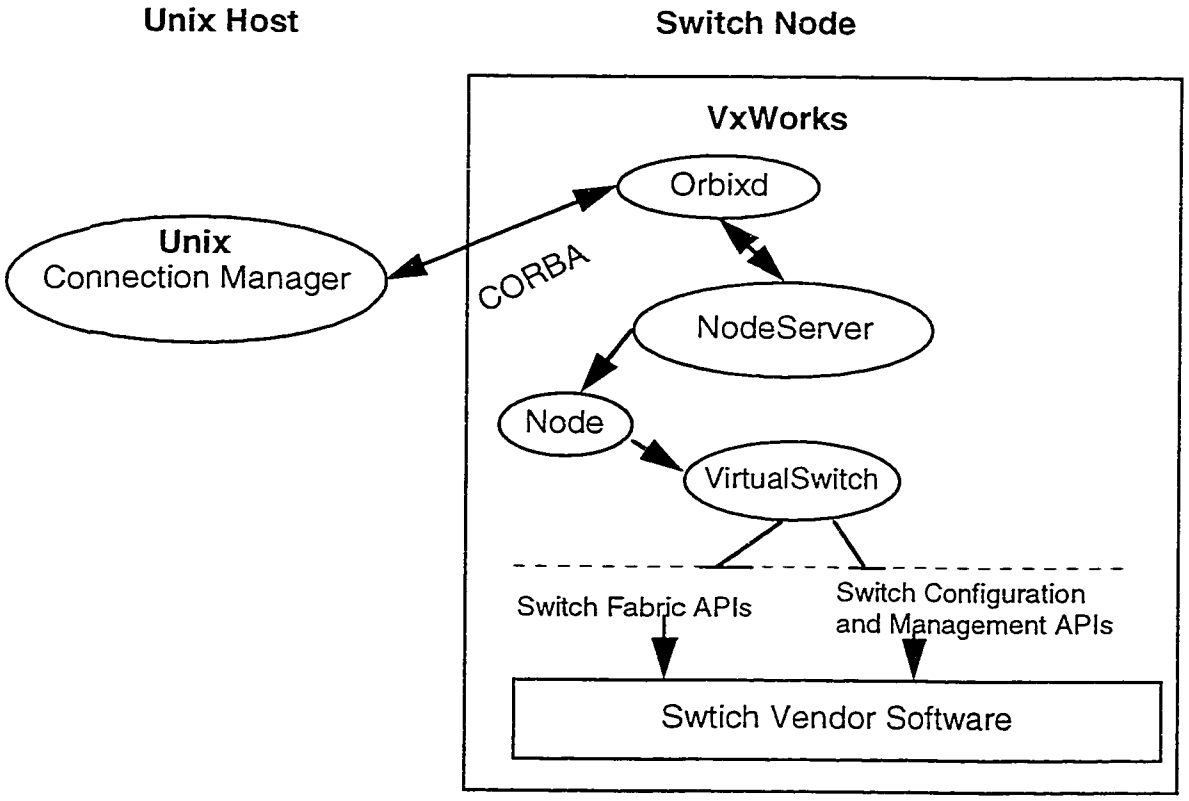


Figure 5.7 An insight look of connection set up in the switch node

In the case of point-to-point connection set up, the CORBA message is composed when the "commitChannel()' method is called by the Connection Manager. Then commitChannel request is sent by the orbix for UNIX daemon running on the worksation to the VIVID ATM switch node. When the CORBA request (or message) is received by the orbix for VxWorks daemon running on the switch node, the orbix daemon task sends it the NodeServer task which is supported by the VxWorks intertask communication mechanism. Subsequently, the commitChannel() method in the VirtualSwitch object is invoked, which in turn calls the addVC() API. When the addVC() API is called, the request finally goes to the VIVID ATM switch software, where according to the desired QoS parameters, the resources of the specified Ingress port and Egress port are checked, if the resources are satisfied, the corresponding Ingress and Egress endpoints VCI/VPI entry will be added to the hardware-level reservation table. Until now, the virtual channel connection is set up. As to the point-to-multipoint connection set up, similar to the point-to-point, the only difference is that the point-to-multipoint set up is performed in different ways by the switch software system.

5.3.3 Test Results

The NodeServer can be tested independently of the Connection Manager running on the workstation. After we implemented the NodeServer, we tested it in the VxWorks environment, including the case of client and NodeServer running on the same switch node and on the different switch node. The NodeServer task can be spawned in two ways: manually or automatically, but before the "NodeServer" is spawned, the "orbixd" i.e. Orbix daemon, must be spawned first. Usually, it is spawned automatically when the switch software system is loaded into the VxWorks target board. When the nodeServer is spawned manually by typing "sp NodeServer" under VxWorks shell, the following messages will be seen on the NMTI terminal screen.

```
==========================================================================

createSwitch is being executed ....
Virtual Switch : vcivpiindextable created ...
Virtual Switch : vcivpiinusetable created ...
Virtual Switch : maxvpivcitable created ...
Virtual Switch : maxvpivcitable created ...
Virtual Switch : inrestable created ...
Virtual Switch : outrestable created ...
Virtual Switch : No of ports = 32
   ..  ..   ..   ..   ..  ..   ..  ..

Switch Node Ip Addr is 138.120.143.25
NextEndPoint Port Vpi Vci is 0 0 5
Virtual Switch : Blocking VCI/VPI/Port 5 0 0
NextEndPoint Port Vpi Vci is 0 0 16
Virtual Switch : Blocking VCI/VPI/Port 16 0 0
NextEndPoint Port Vpi Vci is 0 15 32
Virtual Switch : Blocking VCI/VPI/Port 32 15 0
NextEndPoint Port Vpi Vci is 1 0 5
Virtual Switch : Blocking VCI/VPI/Port 5 0 1
NextEndPoint Port Vpi Vci is 1 0 16
Virtual Switch : Blocking VCI/VPI/Port 16 0 1
NextEndPoint Port Vpi Vci is 1 0 40
Virtual Switch : Blocking VCI/VPI/Port 40 0 1
NextEndPoint Port Vpi Vci is 1 15 32
Virtual Switch : Blocking VCI/VPI/Port 32 15 1
NextEndPoint Port Vpi Vci is 2 0 5
Virtual Switch : Blocking VCI/VPI/Port 5 0 2
NextEndPoint Port Vpi Vci is 2 0 16
Virtual Switch : Blocking VCI/VPI/Port 16 0 2
 ..  ..   ..  ..   ..  ..  ..  ..

[NodeServer: New Connection
(.newbridge.com,IT_daemon,*,VxWorks/Orbix,pid=24535392,,optimised) ]
[orbixd: New Connection
(.newbridge.com,NodeServer,*,VxWorks/Orbix,pid=24455696,,optimised) ]
[orbixd: Registering Persistent Server NodeServer (id = 24455696) ]
[orbixd: Dynamically assigning internet port 1601 ]
[NodeServer: Server "NodeServer" is now available to the network ]
[     Configuration Orbix/1601/Orbix-XDR ]

==========================================================================
```

Figure 5.8 Display message when the NodeServer is spawned

In the automatic approach, the same as "orbixd", the nodeServer is spawned when the switch software system is loaded into the VxWorks target board. In this case, by typing "i" command under VxWorks shell, we can see "orbixd" and "NodeServer" which have existed in the spawned task list on the switch node.

In the following, we give a simple example of point-to-point and point-to-multipoint virtual channel set up within the switch node. The 32 ports of the CS3000 VTVID switch in our implementation is numbered from 0 to 31.

```
#include "switch.hh"
#include <stdio.h>

client (char* host)
{
   ORB_init();
   VirtualSwitch* vsw;
   short  invpi, invci, inport;
   short outvci, outvpi, outport;

   TRY {
      vsw = VirtualSwitch::bind("Switch:NodeServer", host, IT_X);
   }
    CATCHANY {
        printf("Error Happened! \n");
        IT_X.printf();
        ORB_end();
        exit(-1);
    } ENDTRY

   TRY {
    // getOutputChannelIdentifer
     vsw->getOutputChannelIdentifier(Outvci, OutVpi, OutPort, IT_X);
   }
   CATCHANY {
      printf("getOutputChannelIdentifier failed!\n");
      IT_X.printf();
      ORB_end();
      exit(-1);
   } ENDTRY

    TRY {
     //setInputChannelIdentifier
      vsw->setInputChannelIdentifier(InVci, InVpi, InPort, IT_X);
    } CATCHANY {
        printf("setInputChannelIdentifier failed!\n");
        IT_X.printf();
        ORB_end();
        exit(-1);
      } ENDTRY

    TRY {
     // commitChannel
      vsw->commitChannel(InVCi, InVpi, InPort, OutVci,
                         Outvpi, OutPort, VBR, IT_X);
    } CATCHANY {
          printf("CommitChannel failed!\n");
          IT_X.printf();
          ORB_end();
          exit(-1);
      } ENDTRY
}
```

Figure 5.9 A point-to-point commitChannel "client"

For example, to set up a point-to-point connection between endpoint "1-2; 0/40" to "1-3;0/50" and the required QOS service class is VBR , i.e. From Ingress Port 1; VPI = 0; VCI = 40; to Egress Port 2; VPI = 0; VCI = 50; we may have the CORBA "client" as shown in Figure 5.9.

Assume the client and NodeServer run on the same switch node, after the client is compiled and loaded into the VxTarget board, together with NodeServer and orbixd, as long as we type "sp client", the following message will come out on the screen:

```
Virtual Switch : Getting output entry with Port Vpi Vci = 2 0 50
Virtual Switch : Port VPI VCI =  2 0 50 is currently NOT in use.
Virtual Switch : getOutput successful, Port Vpi Vci = 2 0 50
Virtual Switch : setting Input entry with Port Vpi Vci = 1 0 40
Virtual Switch : Input entry added successfully.
commitChannel 1 0 40 -> 2 0 50
Virtual Switch : committing Entry with Port Vpi Vci = 1 0 40 --> 2 0 50
Virtual Switch : Found output reservation  with Port Vpi Vci = 2 0 50
Virtual Switch : output adding entry to vcivpiinusetable !
Switch : Channel P2P committed successfully, Port Vpi Vci = 1 0 40--> 2 0 50
```

Figure 5.10 Display message of point-to-point connection set up between two endpoints

At this moment, to verify the result, we can go to the NMTI level 5, type "CONFIG CONNECT", we can see that the point-to-point connection has been successfully set up (see Figure 5.11). The peakrate of 149760 kb/s is used for VBR class of traffic.

```
VIVID Switch      V42112-Z1-8M                        Alarms:0  No Date 00:10R

Type      Creator Endpoint 1  Endpoint 2     Peak Rate 1->2 Peak Rate 2->1
--------- ------  ----------- ----------     -------------- --------------
P2P  PVC  NMTI     1-2;0/40     1-3;0/50       149760 kb/s    149760 kb/s



CONFIG CONNECT 1-2;0/40

1-SHOW_GROUP     2-DISCONNECT     3-TO_ENDPOINT  4-AS_ROOT     5-TRAFFIC
6-               7-MORE           8-CANCEL       9-QUIT        0-
```

Figure 5.11 Point-to-point virtual connection set up display

When we type "1-Show_Group", all the connections are listed as follows:

```
VIVID Switch      V42112-Z1-8M                          Alarms:0  No Date 00:11R

Type      Creator Endpoint 1      Endpoint 2      Peak Rate 1->2 Peak Rate 2->1
--------- ------  --------------- --------------- -------------- --------------
P2P  PVC  NMTI     1-2;0/40         1-3;0/50        149760 kb/s    149760 kb/s
P2P  PVC  NMTI     1-2;15/32        CPSS               64 kb/s        64 kb/s
P2P  PVC  NMTI     1-3;0/5          SIG                74 kb/s        74 kb/s
P2P  PVC  NMTI     1-3;0/16         ILMI             1587 kb/s      1587 kb/s
P2P  PVC  NMTI     1-3;0/50         1-2;0/40        149760 kb/s    149760 kb/s
P2P  PVC  NMTI     1-3;15/32        CPSS               64 kb/s        64 kb/s
P2P  PVC  NMTI     1-4;0/5          SIG                74 kb/s        74 kb/s
P2P  PVC  NMTI     1-4;0/16         ILMI             1587 kb/s      1587 kb/s
P2P  PVC  NMTI     1-4;15/32        CPSS               64 kb/s        64 kb/s
P2P  PVC  NMTI     2-1;0/5          SIG                74 kb/s        74 kb/s
P2P  PVC  NMTI     2-1;0/16         ILMI             1587 kb/s      1587 kb/s
P2P  PVC  NMTI     2-1;15/32        CPSS               64 kb/s        64 kb/s
P2P  PVC  NMTI     2-2;0/5          SIG                74 kb/s        74 kb/s
P2P  PVC  NMTI     2-2;0/16         ILMI             1587 kb/s      1587 kb/s



CONFIG CONNECT 1-2;0/40 SHOW_GROUP

1-SHOW_CONNECT  2-PAGE_DOWN     3-              4-              5-
6-PAGE_UP       7-              8-CANCEL        9-QUIT          0-
```

Figure 5.12 Display of connections within the current switch node

As to the point-to-multipoint connection, if we want to set up a multicast connection from root endpoint: "1-1; 0/40"; to two leaf endpoints: "1-2;0/50"; and "1-3; 0/60".

Similar to point-to-point connection set up, assuming the client and NodeServer running on the same switch node, after the client is compiled and loaded into the VxTarget board, as long as we type "sp client", the following message we will see on the screen:

```
Virtual Switch : Getting output entry with Port Vpi Vci = 2 0 50
Virtual Switch : Port VPI VCI =  2 0 50 is currently NOT in use.
Virtual Switch : getOutput successful, Port Vpi Vci = 2 0 50
Virtual Switch : setting Input entry with Port Vpi Vci = 1 0 40
Virtual Switch : Input entry added successfully.

commitChannel 1 0 40 -> 2 0 50
Virtual Switch : committing Entry with Port Vpi Vci = 1 0 40 --> 2 0 50
Virtual Switch : Found output reservation  with Port Vpi Vci = 2 0 50
Virtual Switch : output adding entry to vcivpiinusetable !
There is no P2MP connection on the root right now!
Switch : Channel P2MP committed successfully, Port Vpi Vci = 1 0 40--> 2 0 50

Virtual Switch : Getting output entry with Port Vpi Vci = 3 0 60
Virtual Switch : Port VPI VCI =  3 0 60 is currently NOT in use.
Virtual Switch : getOutput successful, Port Vpi Vci = 3 0 60

commitChannel 1 0 40 -> 3 0 60
Virtual Switch : committing Entry with Port Vpi Vci = 1 0 40 --> 3 0 60
Virtual Switch : Found output reservation  with Port Vpi Vci = 3 0 60
Virtual Switch : output adding entry to vcivpiinusetable !
Switch : Channel P2MP committed successfully, Port Vpi Vci = 1 0 40--> 3 0 60.
```

Figure 5.13 Display message of point-to-multipoint connection set up between two endpoints

At this moment, we can go to the NMTI level 5, type "CONFIG CONNECT", we can see the point-to-multipoint connection has been set up (see Figure 5.14).

```
VIVID Switch      V42112-Z1-8L                Alarms:0  No Date 00:45R

Type        Creator          Root Endpoint      Peak Rate Root->Leaf
--------    ------          ---------------    --------------------
P2MP         NMTI               1-2;0/40           149760 kb/s

Leaf Endpoints:
   1-3;0/50      1-4;0/60


CONFIG CONNECT 1-2;0/40


1-SHOW_GROUP  2-DISCONNECT  3-TO_ENDPOINT    4-AS_ROOT     5-TRAFFIC
6-            7-MORE        8-CANCEL         9-QUIT        0-
```

Figure 5.14 Point-to-multipoint virtual connection set up display

When we type "1-Show_group", all the connections within the current node are displayed as followed:

```
VIVID Switch      V42112-Z1-8L                      Alarms:0  No Date 00:46R

Type     Creator Endpoint 1     Endpoint 2     Peak Rate 1->2 Peak Rate 2->1
-------- ------  ----------- ---------------- -------------- --------------
P2P  PVC  NMTI    1-1;0/5         SIG              74 kb/s        74 kb/s
P2P  PVC  NMTI    1-1;0/16        ILMI           1587 kb/s      1587 kb/s
P2P  PVC  NMTI    1-1;15/32       CPSS             64 kb/s        64 kb/s
P2P  PVC  NMTI    1-2;0/5         SIG              74 kb/s        74 kb/s
P2P  PVC  NMTI    1-2;0/16        ILMI           1587 kb/s      1587 kb/s
P2MP PVC  NMTI    1-2;0/40        2 Leaves     149760 kb/s
P2P  PVC  NMTI    1-2;15/32       CPSS             64 kb/s        64 kb/s
P2P  PVC  NMTI    1-3;0/5         SIG              74 kb/s        74 kb/s
P2P  PVC  NMTI    1-3;0/16        ILMI           1587 kb/s      1587 kb/s
P2MP PVC  NMTI    1-3;0/50        1-2;0/40          0 kb/s
P2P  PVC  NMTI    1-3;15/32       CPSS             64 kb/s        64 kb/s
P2P  PVC  NMTI    1-4;0/5         SIG              74 kb/s        74 kb/s
P2P  PVC  NMTI    1-4;0/16        ILMI           1587 kb/s      1587 kb/s
P2MP PVC  NMTI    1-4;0/60        1-2;0/40          0 kb/s


CONFIG CONNECT 1-2;0/40 SHOW_GROUP

1-SHOW_CONNECT  2-PAGE_DOWN     3-              4-              5-
6-PAGE_UP       7-              8-CANCEL        9-QUIT          0-
```

Figure 5.15 Display of connections within the current switch node

5.4 Modifications of Xbind Connection Manager

According to the project proposal, the focus of our work is on the VIVID ATM switch side. As far as the Connection Manager software is concerned, because the intention of our work, to a large extent, is to scale the Xbind system to the real-time environment, we used the code of Connection Manager of the Xbind system. However, considering the difference between the Fore switch system and VIVID ATM switch, some modifications of the source code of Xbind Connection Manager had to be made.

In order to set up connections between two network endpoints, the tasks performed by the Connection Manager mainly include:

- Route Selection. To set up connections between network endpoints, a path between the source and destination must be selected first. Only after the route is known to the Connection Manager, the Connection Manager will in parallel send the "commitChannel" request to all the NoderServers along the route. A RouteObject is used to select a path from the source and the destination endpoint.

- Resource Reservation. Resource reservation mainly refers to reserving and setting switching table entries in the physical switch nodes along the route. These directly use the methods provided by the VirtualSwitch interface, which also constitute the simplest building blocks for the construction of connection management service.

Based on these, let's see how a point-to-point connection between two network endpoints is established in such an "open signaling" environment".

To establish a connection, the Connection Manager first invokes the RouteObject to select a shortest path between the source endpoint and the destination endpoint. After the route, which consists of a list of switch nodes along the path, is available, the Connection Manager distributively and in parallel sends requests to all the switch nodes

belonging to the same path. When the Orbix daemon running on each switch node receives the CORBA request, it delivers the request to the NodeServer, and the corresponding commitChannel method is invoked. Then through the open APIs for the VIVID ATM switch control, the request is transmitted to the internal VIVID ATM switch software system, and finally the hardware-level resources are reserved and the switching table entry is added in the physical switch reservation table. After the virtual channel is committed in each switch node along the route, the Connection Manager will be notified of the success, otherwise, a failure will be returned.

As to the connection removal, with respect to the VIVID software system, we used a different approach from the Xbind system. In Xbind, the ATM switch is modeled as consisting of two parts: Switch Fabric and Switch Link. Correspondingly, in addition to the "Virtualswitch" Interface modeling the functionality of the physical switch fabric for the connection channel setup and removal, a "VirtualLink" interface, which models the functionality of the ports or line access modules of a physical switch, is also provided. The VirtualLink is mainly used to identify the IP address of another end point of the link with which the port exchanges data. As such, during the connection tear down, it is easy to find the adjacent switch node to which the current port is connected and subsequently tear down the connection within the adjacent switch node. However, we are not able to provide such a "VirtualLink" interface over the VIVID software system. That is to say, the NodeServer that we developed over VIVID ATM switches does not provide the capability to get the IP address of the adjacent switch node. Therefore, for the connection removal purpose, the corresponding "route" information has to be cached by the Connection Manager. Due to this, we made some changes to the Xbind Connection Manager source code. When the connection needs to be disconnected, the Connection manager will, according to the cached "route" information, send a CORBA message to the corresponding switch nodes to remove the channels.

Chapter 6

Conclusions

In this thesis, based on "network programmability" and "open signaling" issues, an open signaling system over the VxWorks real-time OS based VIVID ATM switches was presented. The objective of the work was to examine the broadband signaling issues from easier service creation and deployment point of view, and assess the feasibility of using advanced distributed object computing technologies, such as CORBA, for building the high-level connection management service over a real-time environment.

The project that the thesis work was involved is a TRIO Industrially Specified Research Project (ISRP). It aimed at a joint engineering-research project for a Switch Management and Control Software based on an advanced distributed processing environment. According to the proposal, the focus of our thesis work was on the ATM switch side, i.e. development of a switch management and control system over the VIVID ATM switch. To obtain the objective, as we described in Chapter 5, we first deployed a CORBA environment for the VxWorks based VIVID ATM switches, then based on the VIVID ATM switch software system, we developed the open APIs for the VIVID ATM switch control. With the support of the open APIs, we developed the "VirtualSwitch" CORBA Interface that can fully model the functionality of the VIVID ATM switches, and finally the NoderServer was developed and tested.

In accordance with the project proposal, the thesis work has been successfully completed. Although some of our work is based on the Xbind system, due to the uniqueness of the VIVID ATM switch software system, some new challenges were tackled and new features were added. The biggest challenge that we met was the deployment of CORBA/Orbix environment over the VIVID ATM switch system. It took us very long time to figure out how to integrate the IONA's Orbix for VxWorks into the VIVID switch software system. In the case of the new features added, the "VirtualSwitch" Interface for the VIVID ATM switch not only supports point-to-point, but also supports point-to-multipoint virtual channel set up and removal. In addition, by incorporating the QoS class parameter in the commitChannel() methods, it is possible for users to choose the desired QoS class (ABR, VBR, UBR and CBR), so as to meet their specific demands.

With the Switch Management and Control system that we developed for VIVID ATM switches, we intended to scale the Xbind system to the VIVID real-time environment. Therefore, some modifications of the Xbind Connection Manager software

were made. However, for some reasons, we have not tested it yet. Consequently, the first aspect of our future work should focus on testing the modified Connection Manager over the VIVID environment so as to successfully port the Xbind system into the real-time OS environment.

The Xbind software system available can only handle point-to-point connection establishment. Since some new methods were added in the VirtualSwitch interface to handle point-to-multipoint virtual channel set up and removal within the switch node, this makes possible for us to develop a multicast Connection Manager. This should be the second aspect of our future work.

In terms of easier service creation and deployment, "open signaling" can be considered as the next generation signaling approach for future broadband networks. However, compared with the traditional UNI/NNI based Q.2931 protocol, its performance needs to be further investigated. So far, the COMET group has started some work on this area and their research indicates that connection establishment latency for point-to-point call establishment in the Xbind system over the UNIX environment has provided reasonably good performance [CHAN97]. But, it is pointed out that since the operating system of their ATM switches is not real-time, these exists a small probability that the request from the Connection Manager is swapped out while a connection is in progress causing significant degradation. In this aspect, because the real-time operating system provides the high-performance scheduling capabilities, it is believed that in a real-time environment the performance of the "open signaling" system will be better. Therefore, to study the performance of the connection establishment will also be part of our future work.

References

[ACC86] M. J. Accetta, W. Baron, R.V. Bolosky, D. B. Golub, R.F. Rashid, A. Tevanian, M,W. Young, "Mach: A New Kernel Foundation for UNIX Development", In Proceedings of the Summer USENIX Conference, July 1986

[ACA94] A. S. Acampora, "An Introduction to Broadband Networks, LANs, MANs, ATM, B-ISDN, and Optical Networks for Integrated Multimedia Telecommunications", Plenum Press, 1994

[APP93] M. Appledorn, P. Kungm, R. Saracoo, "TMN + IN = TINA", IEEE Communications Magazine, March, 1993

[BER95] H. Berndt, P. Graubmann, M. Wakano, "Service Specification Concepts in TINA-C", TINA-C Core team, Bellcore, http://www.tina.com

[BUB91] R.Bubenick, J. Dehart, M. Gaddis, "Multipoint Connection Management in High Speed Networks", IEEE INFCOM, Apr 1991, 59-68

[BLA95] Uyless Black, "ATM Foundation For Broadband Networks", Prentice Hall PTR, New Jersey, 1995

[BAR93] W. J. Barr, T. Boyd, Y. Inoue, "The TINA Initiative", IEEE Communications Magazine, March 1993

[CAM97] A.T. Campbell, A. A. Lazar, H. Schulzrinne, R. Stadler, "Building Open Programmable Multimedia Networks", The COMET Group, Center for Telecommunications Research, Columbia University, New York, USA.

[CHO95] W. S. Choe, T. J. Geok, Wang Wei Guo, T. S. Woon, "ATM-Based Multi-Party Conferencing System", IEEE Computer Communications, 1995, pp. 13-21

[CHA95a] M. Chapman, H. Berndt, N. Gatti, "Software architecture for the future information market", IEEE computer communications, Vol. 18. No. 11, Nov. 1995

[CHA95b] M. Chapman, S. Montesi, "Overall Concepts and Principles of TINA", TINA Consortium, 1995

[CHAN96] M. C. Chan, J. F. Huard, A. A. Lazar, K.S. Lim, "On Realizing a Broadband Kernel for Multimedia Networks", Third COST 237 International Workshop on Multimedia Telecommunications and Applications, Barcelona, 25-27 Nov. 1996, http://www.ctr.columbia.edu/~publications

[CHAN97] M.C. Chan, "Designing a High Performance TM Connection Management System", Ph.D Thesis, Center for Telecommunications Research, Columbia University, New York, USA, 1997.

[CHU97] P.E. Chung, Y. Huang, S, Yajnil, D. Liang, J. C. Shih, C.Y. Wang, Y.M.Wang, "DCOM and CORBA Side by Side, Step by Step, and Layer by Layer", http://www.akpublic.research.att.com/

[COM91] D. Comer, D. Stevens, "Internetworking with TCP/IP", Prentice Hall, Englewood Cliffs, NJ, 1991

[DAA97] M.Daami, "Synhronization of Control of Coded Video Streams: Algorithms and Implementation", B.A.Sc Thesis, University of Ottawa, 1997

[DEV97] DCAN, Distributed Control of ATM Networks, http://www.cl.cam.ac.uk/Research/SRG/dcan,

[DUP94] F.Dupuy, G. C. Nilsson, Y. Inoue, "The TINA Consortium: Towards Networking Telecommunications Information Services", ISS '95 Contribution, http://www.tina.com

[EDW96] C. Edwards, D. Hutchison, "Approaches to Connection Management within Broadband Networks", Distributed Multimedia Research Group, Computing Department, Lancaster University, 1996

[FUR95] Borko Furt, Milan Milenkovic, "A Guided Tour of Multimedia Systems and Applications", IEEE Computer Society Press, 1995

[FLU95] Francios Fluckiger, "Understanding Networked Multimedia", Addison Wesley, 1995

[GEO96] N. D. Georganas, "Topics in Communications: Multimedia Communictions", ELG 7177 Lecture Notes, Dept. of Electrical Engineering, University of Ottawa, 1996

[HAF97] Abdelhakim Hafid, Gregor von Bochmann, Rachida Dssouli, "Distributed Multimedia Applications and Quality of Service", University of Montreal, 1997

[HER88] F. Herrmann, F. Armand, M. Rozier, M. Gien, M. Guillemont, P. Leonard, S. Langlois, W. Neuhauser, "CHORUS, A New Technology for Building UNIX System", Proc. EUUG Autumn Conference, Cascais, Portugal, pp1-18, October

[HAN94] R. Handel, M. N. Huber, S. Schroo, "ATM Network", Second Edition, Addison-Wesley, 1994

[HOR96] C. Horn, A. O'Toole, "Distributed Object Oriented Approaches", Proceedings of IFIP/IEEE International Conference on Distributed Platforms, 1996

[HYM91] J.M. Hyman, A.A. Lazar, G. Pacifici, "Real-time Scheduling with Quality of Service Constrains", IEEE Journal on Selected Areas in Communications, V0l. 19, Sep. 1991, pp. 1052-1063.

[HYM93] J.M. Hyman, A.A. Lazar, G. Pacifici, "A Seperation Principle between Scheduling and Admission Control for Broadband Switching", IEEE Journal on Selected Areas in Communications, Vol. 11, May 1993, pp. 605-616

[IMA93] HP, IBM and Sunsoft, "Multimedia System Service: Version 1.0", June 1, 1993, URL: http://www.ima.org:80/forums/imf/mss

[ION95] IONA Technologies Ltd. Orbix 2. IONA Technologies Ltd., November 1995, Release 2.0

[ION97] IONATechnologies Ltd. "Orbix for VxWorks 1.3.5 programmer's Guide", 1997

[KAW91] M. Kawarasaki, B. Jabbari, "B-ISDN Architecture and Protocol", IEEE International Workshop on Selected Area in Communications, Vol. 9, No. 9, Dec. 1991

[KIN96] B. Kinane, D. Muldoweny, "Distributed Broadband Multimedia Systems Using CORBA", IEEE Computer Communications, 1996, pp. 13-21

[LAP94] T. F. LaPorta, M. Veeraraghavan, E. Ayanoglu, M. Karol, R. D. Ditlin, "B-ISDN: A Technological Discontinuity", IEEE Communications Magazine, Oct. 1994

[LAZ97] A.A. Lazar, "Programming Telecommunication Networks", Keynote Address at the International Workshop on Quality of Service, Columbia University, New York, 1997, published in the processings of the workshop.

[LAZ96a] "Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture", A. A. Lazar, Koon-Seng Lim, F. Marconcini, Journal of Selected Areas in Communications, Sep. 1996

[LAZ96b] A. A. Lazer, F. Marconcini, "Towards an Open API for ATM Switch Control", COMET Group, Department of Electrical Engineering and Center

for Telecommunications Research, Columbia University, New York, USA. April 1996. URL: http://www.ctr.columbia.edu/comet/xbind/xbind.html

[LAZ96c] A.A. Lazer, K. S. Lim, F. Marconcini, "Xbind: The Application Programmer's Manual", COMET Group, Department of Electrical Engineering and Center for Telecommunications Research, Columbia University, New York, USA. June 1996

[LAZ96d] A.A. Lazer, K. S. Lim, F. Marconcini, "Xbind: The System Programmer's Manual", COMET Group, Department of Electrical Engineering and Center for Telecommunications Research, Columbia University, New York, USA. Sep. 1996

[LAZ96e] A.A.Lazar, K. S. Kim, F. Marconcini, "The Binding Interface Base", COMET Group, Departement of Electrical Engineering and Center for Telecomminications Research, Coulmbia University, March, 1996

[LAZ95a] A. A. Lazar, K. S. Lim, F. Marconcini, "Binding Model: Motivation and Description", CTR Technical Report #411-95-17, Center for Telecommunication Research, Columbia University, New York, June 1995, http://www.ctr.columbia.edu/comet/xbind/xbind.html

[LAZ95b] A.A. Lazar, S. Bhonsle, K .S. Lim, "A Binding Architecture for Multimedia Networks", IEEE Journal of Paralle and Distributed Computing, Vol. 30. No. 2, Nov 1995, pp 204-216.

[LAZ95c] A.A. Lazar, "A Research Agenda for Multimedia Networking", Department of Electrical Engineering and Center for Telecommunications Research, Columbia University, New York, USA, July 1995

[LAZ94] A. A. Lazar, "Challenges in Multimdia Networking", Proceedings of the International Hi-Tech Forum, Osaka '94, Japan, Feb 24-25, 1994. Avaiable at URL: http://www.ctr.columbia.edu/comet/xbind/references.html

[LIM96] Kong Seng Lim. Overview of Xbind, http://www.ctr.columbia.edu/comet/xbind

[MIN93] Daniel Minoli, Robert Keinath, "Distributed Multimedia Through Broadband Communications", Artech House, 1993

[MIN89] S. Minzer, D. Spears, "New Directions in Signalling for Broadband ISDN", IEEE Communication Magazine, Feb. 1989

[MAGE93] T. Magedanz, "IN and TMN: the Basis for Future Information Networking Architectures", IEEE Computer Communications, Vol. 16. No. 5, May 1993

[MAG93] RACE project, R2044 MAGIC-Multiservice Applications Governing Integrated Control, "Signalling Network Architecture", 9th Deliverable, Sep. 1993

[MAG94] RACE project, R2044, MAGIC-Multiservice Applications Governing Integrated Control, "Services and Processing Concepts for B-ISDN Signalling", 13th Deliverable, Dec. 1994

[MAG95] RACE project R2044 MAGIC-Multiservice Applications Governing Integrated Control, 12th Deliverable - Final Report, Apr. 1995

[NAT92] N. Natarajan, G. M. Slawsky, "A Framework Architecture for Multimedia Information Networks", IEEE Communication Magazine, Feb. 1992

[ODP95] "Open Distributed Processing Standard", http://www.dtsc.edu.au /AU/research_news/odp/ref_model/standards.html

[OMG94] OMG, "CORBAservice: Common Object Services Specificaltion", 1994

[OMG95] OMG, "The Common Object Request Broker Architecture and Specification", v.2.0, July 1995

[RAS97] J. Rasanen, P. Koponen, O. Martikainen, "Service Execution Environment for Digital Multimedia", Laboratory of Telecommunication Software and Multimedia, Helsinki University of Technology.

[SIE96] Jon Siegel, "CORBA fundamentals and Programming", John Willey&Sons Inc. 1996

[STA95] W. Stallings, "Data and Computer Communications", fourth edition, Macmillan, Publishing company, New York, 1995

[STE95a] R. Steinmetz, K. Nahrstedt, "Multimedia: Computing, Communications & Applications", Prentice Hall, Upper Saddle River, New Jersey, 1995

[STE95b] R.Steinmetz, "Analyzing the Multimedia Operating System", IEEE Multimedia, Spring, 1995

[SIQ97] Frank Siqueira, "A Framework for Distributed Multimedia Applications based on CORBA and Integrated Service Networks", Ph.D. Project, http://www.cs.tcd.ie/~frank_siqeira

[SCH97a] D.C. Schmidt, "Distributed Object Computing", IEEE Communications Magazine, Feb. 1997

[SCH97b] D.C. Schmidt, A. Gokhale, T. H. Harrison, G. Parulkar, "A high-Performance End System Architecture for Real-Time CORBA", IEEE Communications Magazine, Vol 14, No.2 Feb. 1997

[SIE96] J. Siegel, "CORBA fundamentals and Programming", John Willey &Sons, Inc. 1996

[STEF95] J.B. Stefani, "Open Distributed Processing: An Architecture Basis for Information Networks", IEEE Computer Communications, Vol. 18. No. 11, Nov. 1995

[TIN95] TINA-C, Service Architecture, Version 2.0, Document No. TB_MDC.012_2.0
_94, March 1995

[TU92] S. W. Tu, G. A. Brenner, H. S. Kim, "An Object-oriented Resource Model for Supporting Signaling and Control of Broadband Services", IEEE

International Conference on Communications, pp. 616-621, 322.6.1-322.6.6, June 1992

[VEE95] M. Veeragraghavan, T. La Porta. W. S. Lai, "An Alternative Approach to Call/Congestion Control in Broadband Switching Systems", IEEE Communications, Nov. 1995, Vol. 33, No. 11, pp. 90-97

[VIN97] S. Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environment", IEEE Communications Magazine, Feb. 1997

[VIV97] VIVID CS 3000 Manual NNP 95-1968-01-00-A Issue 1, August 1997

[WAD96] D. Waddington, C. Edwards, D. Hutchison, "Resource Management for Distributed Multimedia Applications", Distributed Multimedia Research Group, Computing Department, Lancaster University, UK. 1996

[WIN93] Wind River System, "VxWorks Programmer's Guide", preliminary Edition, Release 5.1, Feb. 1993.

[WU95] Dakang Wu, "A Survey of Networking Signalling", Department of Computer Science, Washington University, 1995

[WU96] Dakang Wu, "An Efficient Signaling Structure for ATM networks", Department of Computer Science, Washington University, 1996

[YAN95] Z. Yang, K. Duddy, "Distributed Object Computing with CORBA", DSTC Technical Report 23, June 1995

[YUN95a] T. H. Yun, J. Y. Kong, J. W. Hong, "Object-Oriented Modeling of Distributed Multimedia Services", Department of Computer Science and Engineering, Pohang University of Science and Technology, Korea, 1995, http://www.postech.ac.kr

[YUN95b] T. H. Yun, J. Y. Kong, J. W. Hong, "A CORBA-Based Distributed Multimedia System", Department of Computer Science and Engineering,

Pohang University of Science aand Technology, Korea, 1995, http://www.postech.ac.kr

[ZHA93] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource ReServation Protocol", IEEE Network Magazine, September 1993

IMAGE EVALUATION TEST TARGET (QA-3)

150mm

6"

APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989