Validation of Hybrid Multimedia Binding Objects

J. van de Leur

Informatica

RuG

Afstudeerverslag

Validation of Hybrid Multimedia Binding Objects

J. van de Leur

Advisor:
Prof.dr.ir. L.J.M. Nieuwenhuis
Rijksuniversiteit Groningen
Informatica
Postbus 800
9700 AV Groningen

July 1998

Contents

List of Abbreviations

ADSL	Asynchronous Digital Subscriber Line
API	Application Programmer's Interface
ATM	Asynchronous Transfer Mode
AVO	Audio-Visual Object
CIF	Common Intermediate Format
codec	coder/decoder
CORBA	Common Request Broker Architecture
DAVIC	Digital Audiovisual Council
DCOM	Distributed Component Object Model
DMIF	Delivery Multimedia Interaction Framework
DPE	Distributed Processing Environment
DSM-CC	Digital Storage Media - Command & Control
DVD	Digital Versatile Disk
GSTN	General Switched Telephone Network
GUI	Graphical User Interface
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISO/IEC	International Standardisation Organisation / International Electrotechnical Commission
ITU	International Telecommunications Union
ITU-T	International Telecommunications Union - Telecommunication sector
JMF	Java Media Framework
KTN	Kernel Transport Network
LAN	Local Area Network
MPEG	Motion Picture Experts Group
MCU	Multipoint Control Unit
ODE	Open Distributed Environment
ODP-RM	Open Distributed Processing Reference Model
OMG	Object Management Group
OOA&D	Object-Oriented Analysis & Design
PDU	Protocol Data Unit
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RFC	Request For Comments
RFP	Request For Proposals
RSVP	Resource Reservation Protocol
RTP	Real-Time Protocol
RTSP	Real-Time Streaming Protocol
SAP	Service Access Point

SCN	Switched Circuit Network
SDK	Software Development Kit
SDU	Service Data Unit
SFP	Simple Flow Protocol
TCP	Transport Control Protocol
TINA	Telecommunications Information Networking Architecture
TINA-C	Telecommunications Information Networking Architecture Consortium
TINA NRA	TINA Network Resource Architecture
TN	Transport Network
UDP	User Datagram Protocol
UML	Unified Modeling Language
VC	Virtual Channel
VP	Virtual Path
WAN	Wide Area Network

1 Introduction

1.1 Rationale

From both the technical and organisational points of view, it is a general trend that telecommunications and information technology are converging. This trend is most evident in multimedia systems, since these systems in general consist of a set of computer systems interconnected through a telecommunications network.

Both the IT industry and telecommunications industry provide solutions for implementing multimedia systems. However, in the area of 'control and management of multimedia streams' the solutions are often proprietary, focusing on a particular application domain. This results in inworking problems between the various systems.

The Telecommunications Information Networking Architecture Consortium (TINA-C) has made a first step towards a generic approach for the interworking of large heterogeneous multimedia systems. The control and management of multimedia streams is operated through objects and can be used to encapsulate existing solutions. The Object Management Group (OMG) has moved a step further by adopting standard interfaces for control and management objects. A third important player in this area is the International Telecommunications Union (ITU), which has developed the H.32*x* series standards. These standards are used for audio and video conferencing over various types of telecommunications networks.

These developments give rise to issues like how multimedia services can be added to Open Distributed Environments (resulting in an Open Distributed *Multimedia* Environment).

1.2 Objectives and Approach of this Thesis

It is believed that the convergence of the Telecommunications and IT Sectors can be captured in the 'hybrid multimedia binding object'-concept. It is believed [Leydekkers, 1997] that by using this concept, connections can be established between multimedia equipment supporting existing multimedia standards, thereby utilising investments already made in these standards and equipment. The main objective of this thesis is to investigate the advantages and disadvantages of this concept with respect to existing concepts for the realisation of multimedia connections. Emphasis in these analyses is on control aspects of multimedia connections, like quality of service and multiparty connections.

To achieve this objective, it is first investigated how multimedia can be integrated in an open distributed environment. Then, a reference model is developed to analyse and compare multimedia stream binding solutions. This reference model is used to make strength/weakness analyses of three important multimedia stream binding solutions, each originating from a different area. The analyses are used to validate the hybrid multimedia binding object concept, by investigating the interworking aspects of the solutions which were analysed.

A second objective of this thesis is to give a state-of-the-art overview of existing standards and technologies for multimedia communications. This overview is given in the third and fourth Chapter.

1.3 Organisation of the Following Chapters

The thesis consists of eight Chapters. Chapter 1 gives a short introduction to the area of research and presents the objectives and research approach of this thesis.

Chapter 2 provides the 'building blocks' for the following chapters, giving definitions of 'multimedia' and 'multimedia environments', and explaining concepts of distributed processing environments, open distributed environments and the Open Distributed Processing Reference Model, and the relations between these concepts.

Chapter 3 describes the different components and requirements of a multimedia system (e.g. hardware, network transport and control protocols, and QoS management) and the issues and problems that arise when these components and requirements are integrated with an open distributed environment. In this context the concepts of multimedia binding objects and hybrid multimedia binding objects are introduced.

Chapter 4 describes a number of standards and solutions to establish multimedia connections, developed by standardisation consortia and commercial vendors. This Chapter can be skipped by readers who already have knowledge of multimedia standards like the OMG Control and Management of A/V Streams, TINA Network Resource Architecture and ITU-T H.323.

In Chapter 5 a reference model is developed for the comparison of multimedia stream binding solutions. The purpose of this reference model is to make strength/weakness analyses of multimedia stream binding solutions, which can be used for the comparison and evaluation of multimedia stream binding solutions. The analyses are made from the ODP-RM Information and Computational viewpoints.

In Chapter 6 the TINA NRA, OMG Control and Management of A/V Streams and ITU-T H.323 standards are analysed using the reference model developed in Chapter 5. These analyses are then used to validate the hybrid multimedia binding object concept, by discussing a number of co-operation scenarios.

The complete analyses of these solutions are described in a separate 'Research Note'. This is a separate document which can be obtained at request.

In Chapter 7, the design and implementation of a simplified OMG A/V Streams compliant endpoint component is described, validating a possible scenario of using the hybrid multimedia binding object concept and the OMG Control and Management of A/V Streams specification.

Chapter 8 presents the conclusions and gives recommendations for further study.

2 Multimedia and Open Distributed Environments

2.1 Introduction

This chapter provides the 'building blocks' for the following chapters of this thesis. First, definitions of multimedia and multimedia environments are given. After this, Distributed Processing Environments (DPEs) and Open Distributed Environments (ODEs) are described. An introduction to the Open Distributed Processing Reference Model (ODP-RM) is given, which is a meta standard to describe ODEs.

2.2 What is multimedia?

The word 'multimedia' is a contraction of the two words 'multi' and 'media'. 'Media' refers to types of information or types of information carriers. There are numerous types of media available, for example text, audio, video, and so on. These media can be classified into two classes: *static media* and *dynamic media*.

Static media do not have a time dimension, their contents and meanings do not depend on the presentation time. Examples of static media are text, images, and graphs. Dynamic, or time continuous media, on the other hand do have a time dimension. Their meanings and correctness depend on the rate at which they are presented, and change during the presentation time. Examples of dynamic media are audio, video and animation. Because of the continuous character, dynamic media are mostly referred to as *multimedia streams*. A multimedia system is defined a follows:

> **Multimedia system:** *a system that is capable of handling at least one type of continuous data in digital form as well as static media [Lu, 1996]*

Note that in this definition, the information has to be stored in digital form. This prevents, for example, a VCR to be called a multimedia system (because a VCR records and plays back analogous data). A DVD (Digital Versatile Disk) player instead, is a multimedia system, because the data is stored digitally. The different aspects of a multimedia system are discussed in detail in Chapter 3.

2.3 Multimedia Environments

Figure 2-1 shows a typical multimedia environment. It consists of two multimedia systems (endpoints), connected through a computer network. One system has a camera and a microphone, and acts as the 'source'. The other system has a display and speakers and acts as the 'sink'. The multimedia information flow, or *multimedia stream*, is directed from the source to the sink.

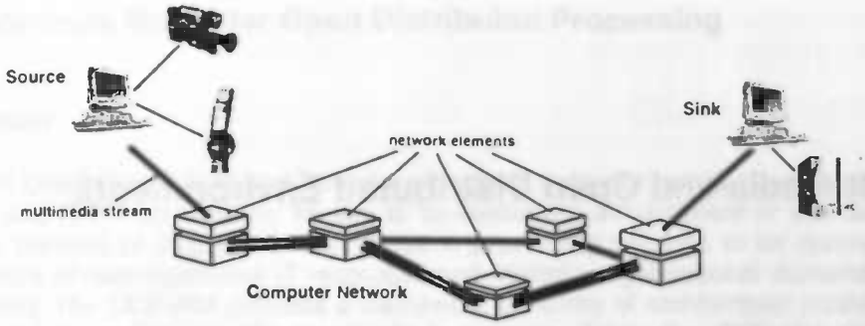


Figure 2-1: Example of a multimedia environment

A multimedia environment is not restricted to two endpoints. In theory, a multimedia environment can consist of an unlimited number of endpoints, where each endpoint can act as both a source and a sink. In practice however, there are certain limits to the number of endpoints, because of limited processing capacity of the hardware and network used.

A multimedia environment often is *heterogeneous*, which means that it can consist of heterogeneous endpoints (consisting of different types of hardware devices, like cameras, telephones, microphones, etc.), connected through heterogeneous networks, which use heterogeneous protocols to send heterogeneous multimedia streams over the network. Figure 2-2 shows a heterogeneous multimedia environment.

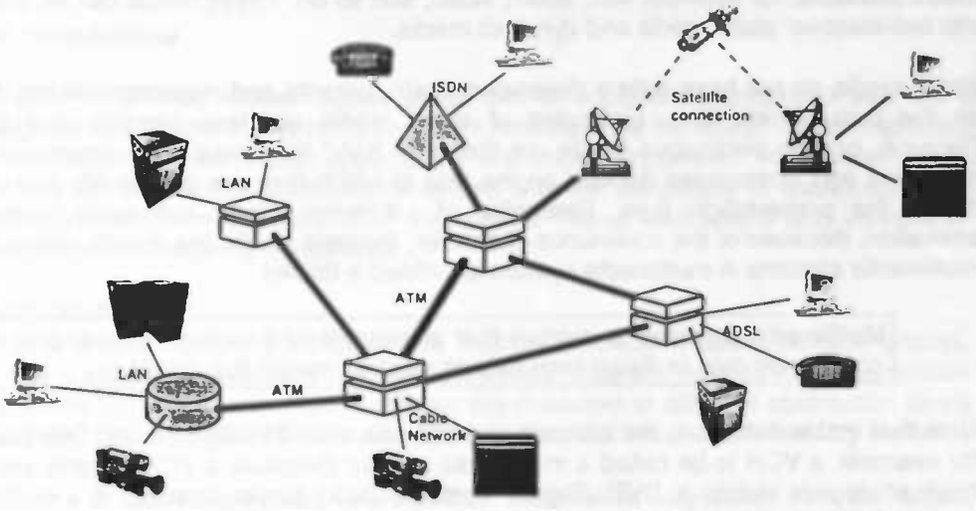


Figure 2-2: Example of a heterogeneous multimedia environment

In a multimedia environment, a connection between devices of different types can in some cases be established. For example, a videoconferencing device can in some cases be connected to a telephone. Because a telephone does not provide video input and output functionality, such a connection consists of only a sound stream. Whether a connection between devices of different types can be established depends on the *codecs* (coder/decoder, the part of the device that codes and decodes the information to be transmitted) and the protocols used.

2.4 Open Distributed Environments

In the mid-sixties, computer systems consisted of terminals connected to monolithic mainframes. In the eighties, the introduction of the PC led to client-server systems, with applications running on the clients and services like printer services and database services running on the server. This resulted in semi-isolated LANs. In the beginning of the nineties companies started to interconnect the LANs of their offices to create one corporate network. These developments lead to increasing complexity of the network infrastructure of a company, due to the heterogeneous hardware, operating systems and

software used in most companies. These heterogeneous environments were a source for inworking problems between the various interconnected systems.

To solve these compatibility problems, and to make better use of the potentials offered by a large number of interconnected computer systems (e.g. dividing tasks over several computers, thereby making use of the processing capacity of all the systems involved), Distributed Processing Environments, or DPEs, were developed.

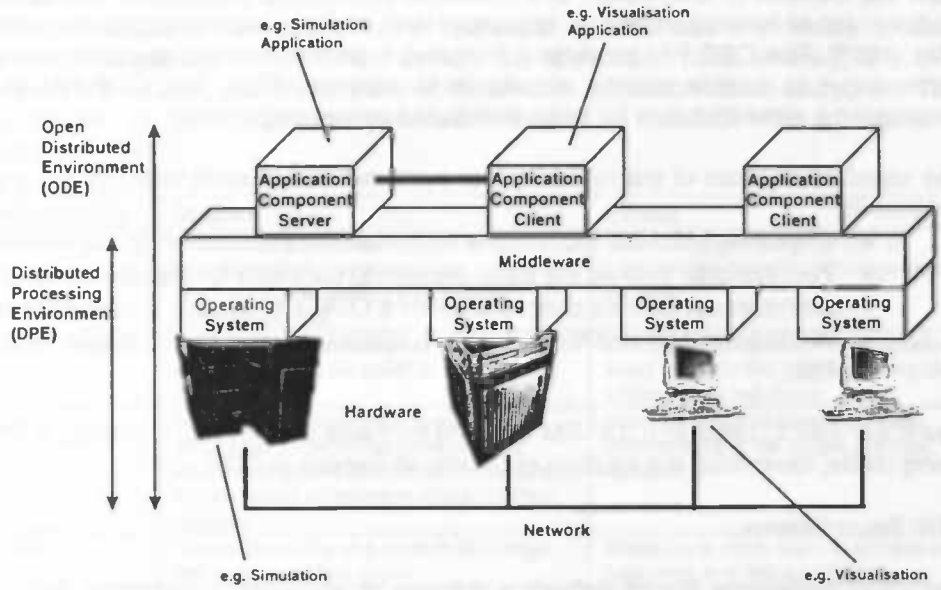


Figure 2-3: Architecture of an Open Distributed Environment with a Simulation/Visualisation application running in it

A *Distributed Processing Environment* (DPE) is the combination of computer systems (running various operating systems), the network connecting them, and *middleware*. Middleware is the software that enables interoperability between application components which are physically distributed over several computing systems. Middleware makes the application code independent from the distributed system. It is called 'middleware' because this software resides between the Operating System and the Applications.

An *Open Distributed Environment* (ODE) uses the facilities of a DPE to create an environment in which application components can (1) be executed independent of the used hardware, operating system, and network technology, and (2) interwork with other application components possibility residing in different computing systems, without modification of the components.

An example of the use of an ODE is shown in Figure 2-3: a visualisation application is divided into a Server component, which physically runs on a computer system, specialised in running simulations, and a Client component, which carries out the visualisation. This component also runs on a specialised visualisation system. All communication and data transfer is carried out via the DPE, in a way that the Client component is not aware of the physical location and implementation of the Server component.

A number of middleware solutions have already been developed (e.g. Corba, DCOM), and it is expected that more will be developed in the future, all with different capabilities and using various standards. To co-ordinate the development of these standards, the ISO/IEC and ITU-T standardisation committees have created a framework which covers all relevant aspects of distributed systems. This framework is called the Open Distributed Processing Reference Model [ODP-RM 1-4, 1995], and is discussed in the next section.

2.5 The Reference Model for Open Distributed Processing

2.5.1 Introduction

The Open Distributed Processing Reference Model is a *de jure* standard, produced by the ISO/IEC and ITU-T committees. Its aim is 'to enable the development of standards that allow the benefits of distribution of information processing services to be realised in an environment of heterogeneous IT resources and multiple organisational domains' [ODP-RM1, 1995]. The ODP-RM provides a framework (in terms of architectural concepts and terminology) to enable specific standards to emerge. Thus, the ODP-RM should be considered a *meta-standard* for open distributed processing.

The standard consists of the following four parts, which are each described in a separate document:

- *Overview*: provides motivations and a tutorial introduction to the main concepts
- *Foundations*: defines the basic modelling concepts for distributed systems
- *Architecture*: defines concepts which a ODP-RM system should possess
- *Architectural Semantics*: provides a formalisation of the concepts behind ODP-RM

See [ODP-RM 1, 1995] to [ODP-RM 4, 1995] for these documents. Currently, a fifth part is being made, describing the addition of Quality of Service in ODP.

2.5.2 ODP Foundations

The ODP Reference Model defines a number of important foundations that are used throughout the standard. The most important foundation is the use of *object orientation* for the specification of distributed applications and their components. In addition, the ODP-RM uses two abstraction mechanisms to deal with the complexity of distributed systems: *distribution transparencies* and *ODP Viewpoints*. These foundations and viewpoints are discussed in the following paragraphs.

Object Orientation
Object Orientation is the concept to use objects to model problem domain entities. An object is a self-contained entity that consists of both data and operations to manipulate the data. In the ODP reference model, system are modelled at different abstraction levels (or viewpoints) by using sets of interacting objects. An object is characterised by the following items:

- *Encapsulation and abstraction*: information contained in an object is *encapsulated*, that is, accessible only through interactions at interfaces supported by the object, thus providing the effect of *abstraction*, by implying that internal details of an object are not visible to other objects.
- *Behaviour/State:* the *behaviour* of an object is defined as the set of all potential actions in which an object may take part. *State* characterises the situation of an object at a given instant.
- *Interfaces:* an *interface* is the only means to access an object. An interface consists of a set of interactions, which can be signal, stream or operational interfaces. An ODP object can have multiple interfaces, possibly of different types.
- *Type and Class:* a *type* is a property of a collection of objects. A *class* is the collection of all objects associated with a given type.
- *Polymorphism:* the property that the same operation can do different things depending on the class that implements it. Objects belonging to different classes can receive the same request but react in different ways. The initiator is not aware of this difference; the receiver interprets the operation and provides the appropriate behaviour.

- *Inheritance:* the mechanism to create subclasses from a parent class, which inherit operations and properties from the parent class. Child classes can add or override operations and properties to define new behaviour. The behaviour of the parent class is not affected by such modifications.
- *Templates:* used to describe common features of objects of the same type. A template contains sufficient information to instantiate a new object from it.

Distribution Transparencies

Distribution transparencies are a set of concepts which make it possible to develop applications independent of on which system the application runs, where (parts of) the application are located, how these parts communicate with each other, and so on. Table 1 shows the set of distribution transparencies defined in the ODP-RM (adapted from [Leydekkers, 1997]).

Transparency	Masks	Effect
Access	Masks the difference in data representation and invocation mechanisms to enable interworking between objects	Solves many of the problems of Interworking between heterogeneous systems
Failure	Masks the failure and possible recovery of other objects (or itself) to enable fault tolerance	The designer can work in an idealised world in which the corresponding class of failures does not occur
Location	Masks the distribution in space of interfaces. Location transparency for interfaces require that interface identifiers do not reveal information about interface location	Provides a logical view of naming, independent of the actual physical location
Migration	Masks the ability of a system to change the location of that object	Migration is often used to achieve load balancing and reduce latency
Relocation	Masks the relocation of an interface from other interfaces bound to it	Allows system operation to continue when migration or replacement of objects occurs
Replication	Masks the use of a group of mutually behaviourally compatible objects to support an interface	Enhances performance and availability of applications
Persistence	Masks the deactivation and reactivation of other objects (or itself) from an object	Maintains the persistence of an object when the system is unable to provide it with processing, storage and communication functions continuously
Transaction	Masks the co-ordination of activities amongst a configuration of objects to achieve consistency	Provides consistency guarantees about interactions between applications

Table 1: Distribution Transparencies

2.5.3 ODP Viewpoints

The ODP reference model uses viewpoints to deal with the complexity of a distributed system. A viewpoint is a representation of a system with the emphasis on a specific concern while ignoring other characteristics that are irrelevant for that viewpoint. Each viewpoint represents a different abstraction level of the original system. ODP uses five viewpoints: the enterprise-, information-, computational-, engineering- and technology viewpoint.

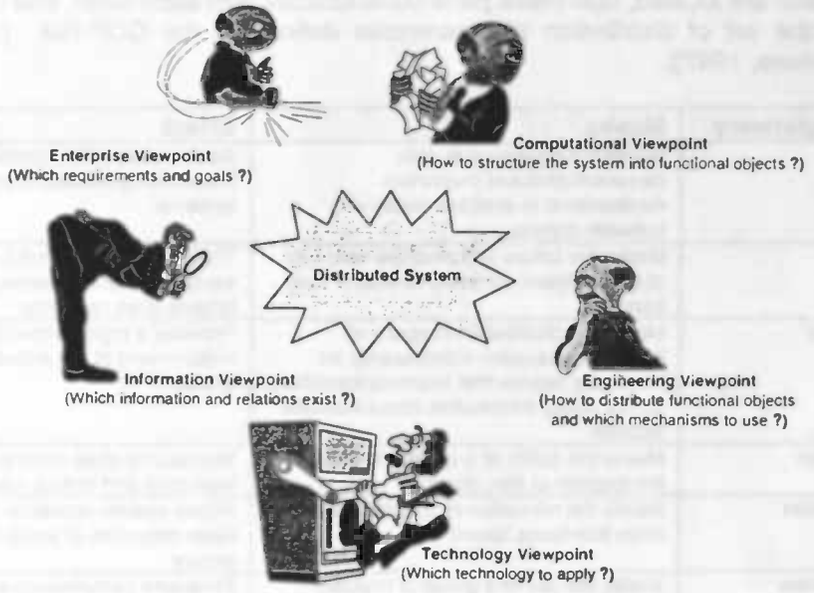


Figure 2-4: ODP-RM Viewpoints

The Enterprise Viewpoint
The Enterprise Viewpoint focuses on the requirements, purpose and policies that apply to the specified system independent of distribution aspects that might be applicable to the system. It covers the business aspects and the human user roles with respect to the system and the environment with which the system interacts. From the Enterprise Viewpoint the overall objectives of an ODP system are seen.

The Information Viewpoint
The Information Viewpoint is concerned with the information that needs to be stored, exchanged and processed in the system of concern. The Information Viewpoint describes the information model of the system and of the individual components identified. It provides a common view, which can be referenced by the specifications of information sources and sinks and the information flows between them.

The Computational Viewpoint
The Computational Viewpoint is concerned with the description of the system as a set of interacting objects, and describes how distributed applications and their components are structured in a distribution transparent way. This implies that the structuring of applications is independent of computers and networks on which they run. This viewpoint specifies the individual, logical components, which are the sources and sinks of information. The model used in the Computational Viewpoint is object based; a distributed application consists of a collection of computational objects.

The Engineering Viewpoint
The Engineering Viewpoint focuses on the infrastructure required to support distributed processing. It is concerned with the distribution mechanisms and the provision of the various transparencies needed to support distribution.

The Technology Viewpoint

The Technology Viewpoint focuses on suitable technologies to support the implementation aspects of the distributed system. It is concerned with the implementation details of the components from which the distributed system is constructed.

2.5.4 Binding

The ODP Reference Model uses the concept of *binding* to describe a communication path between two interfaces. A binding can be either *implicit* or *explicit*. An implicit binding is set up automatically by a DPE to facilitate communication between interfaces. No external action is required to set up such a binding. An explicit binding, on the other hand, is set up after an explicit external request. Another difference between these types of bindings is that an explicit binding is modelled by a special binding object, offering interfaces to control the binding. An implicit binding can not be controlled externally.

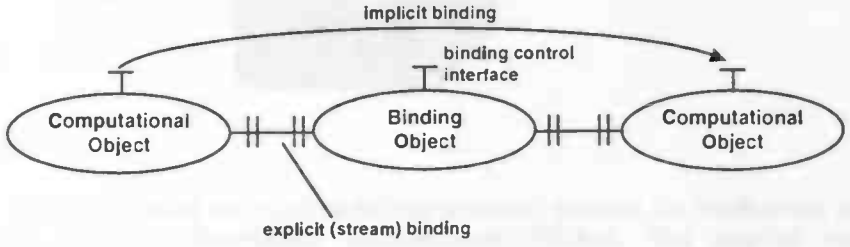


Figure 2-5: Implicit and explicit bindings

The advantage of explicit binding is that a binding can be controlled after it is established. This is especially important in multimedia applications, when change of QoS is required, or when parties are added to or removed from the binding.

2.6 Summary

In this Chapter, a multimedia system is defined as 'a system that is capable of handling at least one type of continuous data in digital form as well as static media'. Multimedia systems are often interconnected through a computer network, resulting in a multimedia environment.

Multimedia environments are usually heterogeneous, consisting of different kinds of hardware, networks, operating systems and software, using different standards. This heterogeneity often causes inworking problems between the various multimedia systems. An Open Distributed Environment is a concept which solves these inworking problems by using the features offered by a Distributed Processing Environment.

The Open Distributed Processing Reference Model (ODP-RM) is a standard which 'enables the development of standards that allow the benefits of distribution of information processing services to be realises in an environment of heterogeneous IT resources and multiple organisational domains'. The ODP-RM defines five viewpoints which each emphasise on specific characteristics of a system: the enterprise, information, computational, engineering and technology viewpoints. The concept of 'binding' is introduced to describe a communication path between interfaces.

3 Concepts of Distributed Multimedia Systems

3.1 Introduction

With the increasing need for multimedia applications, support for multimedia should be incorporated in Open Distributed Environments (ODEs). The special, continuous (streaming) character of multimedia data however, poses a number of requirements on both the design of open distributed environments and on the used technologies, like hardware, network technologies, network protocols, operating systems and software. It is possible that the design of an open distributed environment has to be significantly changed to support multimedia applications.

This chapter discusses the issues and problems which rise when multimedia is incorporated in open distributed environments. Addition of multimedia capabilities to an ODE would in one way significantly extend the capabilities and services of an ODE, resulting in an Open Distributed *Multimedia* Environment. In the other way it would extend the capabilities of multimedia applications because functionality offered by ODEs (e.g. location and hardware transparency) becomes available for these multimedia applications.

The purpose of this chapter is to get a good understanding of the different components of a multimedia system, and the issues and problems that arise when these components are integrated with an open distributed environment.

The structure of the chapter is the following: first, two paradigms used to design (multimedia) systems are introduced. Then, the requirements of hardware and protocols for multimedia systems are discussed, detailing the concepts described in Chapter 2. These requirements are divided in hardware and network technologies, transport and control protocols, quality of service management and multiparty connections. Next, it is discussed what issues and problems arise when multimedia and open distributed environment are integrated. The last section discusses the concept of the *multimedia binding object*, which abstracts from a multimedia communication path in an open distributed multimedia environment.

3.2 The Object-centered and Protocol-centered Paradigms

Open Distributed Environments designed according to the principles of the ODP are designed using the *object-centered paradigm* [Sinderen, 1997]. In this paradigm, system parts are objects, such that the model of a distributed system to be built consists of a collection of interacting objects (see Figure 3-1). The interaction means between objects in this paradigm normally supports a limited set of communication patterns, related to so called interface types, like operation interfaces. The objects in such a system are capable of knowing each others interfaces, so that unambiguous understanding of information exchange is achieved.

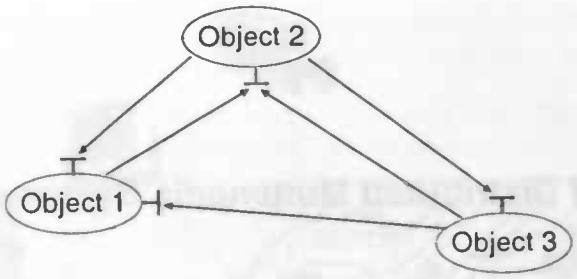


Figure 3-1: Objects interacting through interfaces (object-centered paradigm)

The object-centered paradigm originates from the distributed computing area. The telecommunications area on the other hand, has a strong focus on networks and protocols for transporting data over those network, which are usually designed with the *protocol-centered paradigm* [Sinderen, 1997]. In this paradigm, system parts are protocol entities, and the system as a whole provides a service (see Figure 3-2). The interaction means between protocol entities is a lower level service. Protocol entities communicate with each other by exchanging Protocol Data Units (PDUs), which define the syntax and semantics for unambiguous understanding of the information exchanged between the protocol entities. The model of a system to be built using the protocol-centered paradigm consists of a collection of layered protocol entities, generating a protocol stack.

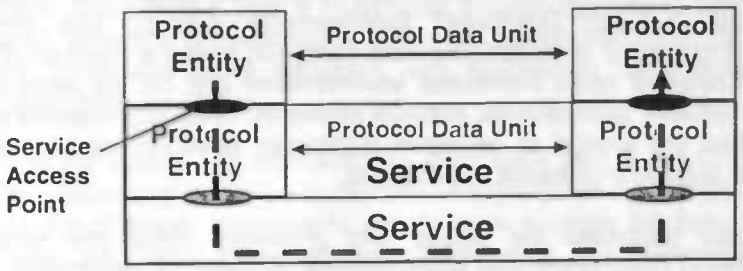


Figure 3-2: Protocol-stack (protocol-centered paradigm)

For the actual transfer of the PDUs, the protocol entities use the Service Access Points (SAPs) provided by the lower-level service. The PDUs are 'encapsulated' in Service Data Units (SDUs), which is the interaction means of the lower-level service.

With the converging telecommunication and information technology areas, distributed computing concepts are more and more used in telecommunications services. An especially important area of services is that of interactive, multimedia network facilities, offered to end-users and integrated in distributed applications. These developments cause that systems are designed using both the object-centered and protocol-centered paradigms, integrating these two disciplines.

3.2.1 Comparing the Object-centered and Protocol-centered Paradigms

As will be described in the rest of this thesis, the advantage of the object-centered paradigm is that interactions between objects are easy to achieve, usually by invoking a method on the peer object. This advantage however has its price in performance: an object interaction usually needs to be converted into PDUs, transmitted by the DPE, and again be converted to restore the original object interaction. The simple way to accomplish object interactions makes this approach very suitable for interactions which require a relatively small amount of complex data, like interactions for high-level binding control and management and configuration negotiations.

Interactions between protocol entities are more difficult, because the facilities offered by a DPE are not available. Information transfer between protocol entities is however more efficient. This makes the protocol-centered approach very suitable for interactions which require a large amount of relatively simple data, like transport of multimedia data.

An interesting remark with respect to the object-centered paradigm is that it is somehow dependent on the protocol-centered paradigm: interactions between objects are

supported by a DPE which internally uses a network infrastructure and protocol-centered standards to create communication paths to accomplish data transfer.

The object-centered and protocol-centered paradigms are used in this thesis to distinguish between standards and technologies for multimedia communications in open distributed environments, which originate from both the distributed computing and telecommunications areas. In the following Chapters it is investigated how a *hybrid multimedia binding object* can be constructed by co-operation of these standards and technologies.

3.3 General Requirements of Multimedia Systems

The capability to transport multimedia data to other computer systems opens important new usages like video-conferencing, video on demand, tele-learning and tele-medicine. However, the large amounts of data involved with multimedia applications, and the special, time-continuous character of multimedia data pose special requirements on computer systems and networks used for multimedia applications. A multimedia system should meet the following design goals [Lu, 1997]:

- The system should have sufficient resources to support multimedia applications
- The system should be able to use available resources efficiently
- The system should be able to guarantee application QoS requirements
- The system should be scaleable

This section describes how these goals can be met. First the required resources and technologies, like hardware issues, codecs, and network technologies are briefly discussed. Then techniques to use these resources and technologies efficiently, like network control protocols and quality of service management protocols are discussed. Also, a section devoted to the special issues involved with multiparty connections (i.e. connections with three or more parties) is included.

3.3.1 Hardware

Multimedia applications need powerful hardware to process the large amounts of data involved with multimedia applications, and to perform the complicated computations needed for (de)coding multimedia data. Figure 3-3 shows components of a computer systems which are important for multimedia applications.

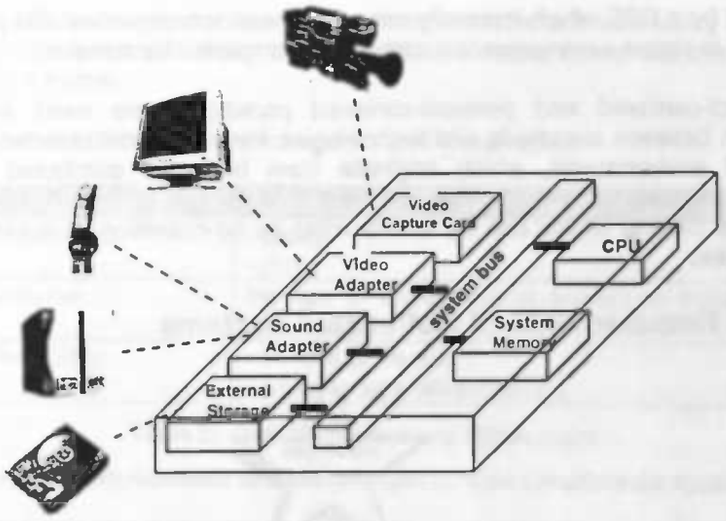


Figure 3-3: Hardware components in a multimedia system

Hardware Components

The most important part in a computer system is the *CPU*, which is used for all computation tasks. It is obvious that a multimedia system requires a powerful CPU to process the large amounts of multimedia data. To relieve the CPU, other hardware components are more and more equipped with specialised chips which take over computational tasks from the CPU.

The other parts of the system are connected to each other and to the CPU through the *system bus*. The system bus is often a bottleneck in a computer system because all data moving between the different components has to pass through the bus. To solve this problem, solutions like direct links between components (e.g. between the system memory and the video adapter) are often used.

Multimedia data mostly consists of images, video and/or sound. Video is usually captured by a specialised *video capture card*. This device uses specialised hardware to code the captured video, thereby releasing the CPU of this task. The *video adapter* facilitates the display of data. Some video adapters are equipped with specialised hardware to decode video data. The *sound adapter* is used to capture and play sound. This functionality is usually integrated on one device, because the hardware needed to code and decode sound is much simpler and therefore cheaper than hardware to code and decode video.

3.3.2 Computer Networks

Most multimedia systems are connected to a network (e.g. a LAN or a WAN). Also, more and more LANs are interconnected to form wide area networks (WANs). This interconnection is usually accomplished by using the IP protocol.

Most network technologies used nowadays are designed to facilitate reliable data transport between computer systems. Multimedia applications however, pose other requirements on a computer network. In the first place, a computer network should be able to transport large amounts of data. Another important requirement is that the data is received at a constant rate, without too much delay (the time between sending and arrival) and jitter (the variation in time between the arrival of for example different frames in a video connection). With today's networks, these requirements are very hard to fulfil. On the contrary, the reliability requirements are not as high as for traditional applications because the loss of for example a video frame does in most cases not largely affect the perceived quality of information. So in short, for multimedia communications, receiving data in time is more important that receiving the data correctly.

It is expected [Wolf, 1997] that the problems encountered in traditional computer networks can largely be solved by adding resource reservation and other Quality of Service

capacities. However, today's computer networks mostly use 'best effort' techniques to transport data, which makes it very difficult to add resource reservation capabilities. In Section 3.3.3 a number of solutions for this problem are discussed.

3.3.3 Quality of Service Management

Quality of Service management is a very important topic in computer networking and multimedia communications. It is implemented by a collection of techniques to guarantee that data is delivered to another computer system over the network correctly, and on time. QoS is generally expressed in terms of, for example, 'amount of delay' or 'bandwidth'. In multimedia applications, QoS management is especially important to deliver data on time, because its time-continuous character makes multimedia data extra sensitive for the delay between the transmission and receiving of data, or changes in this delay.

Reasons to add QoS management to a computer network is that network resources can be utilised more efficiently so less resources are wasted. Also, telecommunications companies and Internet Service Providers which manage the network facilities, can charge users for a certain quality of service provided (this will however only work if the user can be assured that he gets the Quality of Service paid for).

In this Section, and throughout this thesis, the emphasis is on Quality of Service for multimedia in distributed environments. As is shown in the remainder of this Section, multimedia in distributed environments pose a number of additional issues on Quality of Service management, compared with QoS management for multimedia.

First, a definition of Quality of Service for multimedia systems is given, following [Lu, 1996]:

> **Quality of Service:** *is a quantitative and qualitative specification of an application's requirements, which a multimedia system should satisfy in order to achieve desired application quality*

Quantitative aspects are generally expressed in terms of 'the number of frames per second' in a video connection, or 'the audio sample rate' in an audio connection. They are exact values which can be objectively measured. Qualitative aspects however, are much more subjective, and mostly determined in terms of the perceived quality. They are generally expressed in terms of 'video quality', which may be poor in a low resolution, low frame-rate connection, or high in a high resolution, high frame-rate connection. Generally, most qualitative aspects can be translated into objective, quantitative aspects. The QoS categories and dimensions discussed below are all examples of quantitative aspects.

QoS Categories and Dimensions

Quality of Service requirements can be classified into the following categories. Blair & Stefani [Blair, 1998]:

- *Timeliness* - this category contains dimensions related to end-to-end delay of either continuous media or discrete interactions. These dimensions are especially important in interactive multimedia applications, like videoconferencing. Keywords in this category are:
  - *latency* or *delay*, measured in milliseconds and defined as the time between the sending and the arrival of a (part of a) multimedia message, and
  - *delay jitter*, measured in milliseconds and defined as the variation in delay during the transmission

- *Volume* - this category contains dimensions that refer to the throughput of data. For multimedia streams this can be measured in terms of individual elements delivered per second (for example, the throughput of a video stream is measured in frames per second).

- *Reliability* - This category contains dimensions that refer to the reliability of interactions in a multimedia system. This can be measured in terms of frame rate loss or bit error rates within a frame.

Table 2 summarises these different categories and dimensions:

QoS Category	Dimensions
Timeliness	- delay - jitter
Volume	bit rate or throughput in frames or bytes per second
Reliability	- % loss of frames - bit error rate within frames

Table 2: QoS Categories and Dimensions

Table 3 shows QoS requirements and parameters for some multimedia applications:

User Specification	Maximum Delay	Bit Rate (Volume) compressed-uncompressed	Maximum Packet Loss Rate
Phone voice quality	150 ms	16 - 64 kbit/s	1 %
CD quality audio	150 ms	128 - 1440 kbit/s	0.01 %
TV quality video	250 ms	100 Mbit/s	0.1 %
HDTV	250 ms	10 - 800 Mbit/s	$1.0 * 10^{-9}$ %

Table 3: QoS Requirements and Parameters for some multimedia applications

Levels of Quality of Service

In most cases a user can specify in which degree a certain Quality of Service is met. In general three of such levels can be identified:

1. *Deterministic* - The requested QoS must be met 100% in all cases. This guarantee is most expensive in a worst case situation. It can be realised by reserving all needed resources during the connection, even when they are not fully utilised.
2. *Statistical* - The parameters of the requested QoS should not differ more than a certain specified percentage from the original value of the QoS parameters. In practice, the requested QoS is given by specifying a value for the QoS and a percentage of deviation that is still acceptable. The actual reservation of resources can be accomplished by statistically predicting the needed resources at a certain moment. This technique is desired for multimedia applications, because it provides the most efficient trade-off between a guaranteed QoS and efficient use of resources.
3. *Best effort* - No guarantee is provided; resources are used whenever they are available. This technique is used on most LANs, and on the Internet (For example, in The Netherlands it can be perceived through the response time and transmission speed of the Internet when 'America wakes up' !).

Different guarantees can be used for different dimensions of QoS parameters. For example, in a video connection the throughput can be set to a deterministic guarantee, while the delay jitter can be set to a statistic guarantee, specifying an rate of 5% ± 2%. In an audio connection the delay jitter is of much greater importance, so in such a connection the delay jitter may be set to a deterministic guarantee.

A problem not solved yet is how to offer a range of Quality of Service levels in terms of qualitative aspects with connection costs varying per level. In this case, a user may select QoS level 6 on a scale of 1 to 10, independent of the application he/she is going to use, and is billed afterwards for using this QoS level. How to select these QoS levels is still a point of discussion, because with a large amount of applications requiring different QoS parameters, it is very difficult to create a set of levels that can be generally used.

Realisation of QoS

In theory, specification of Quality of Service is quite obvious. In practice however, implementation of QoS management in today's computer networks is a very difficult task. The primary reason for this is that the network technologies and transport and control protocols used in these networks were developed with transmission of discrete data in mind. To incorporate QoS management, in most cases the hardware devices of which the networks are built (like routers, bridges, etc.) have to be adapted, because to guarantee an end-to-end quality of service, all components on the network path between sender and receiver have to be able to reserve the requested resources. The adaptation or replacement of these network components is usually a very expensive task.

It is also important to use co-operating technologies for QoS management, because otherwise network elements that use different technologies can not co-operate to offer an end-to-end quality of service. This is of special importance on the Internet, because of the heterogeneous nature of this network.

To overcome these problems, a number of protocols to implement QoS are developed. These protocols are discussed in Section 3.4.

3.3.4 Multiparty Connections

Another important topic in multimedia communications is the ability to establish multiparty connections. Multiparty connections are defined as 'connections between three or more parties simultaneously'. By adding facilities for multiparty connections, the capabilities of a multimedia system are largely extended.

Two types of multiparty connections can be identified: point-to-multipoint and multipoint-to-multipoint communications. The first type is essentially a subset of the second set, but it is mentioned separately because in point-to-multipoint connections the data usually travels from one sender to a (possibly large) number of receivers. This type of connection is also called broadcasting. See also Figure 3-4.

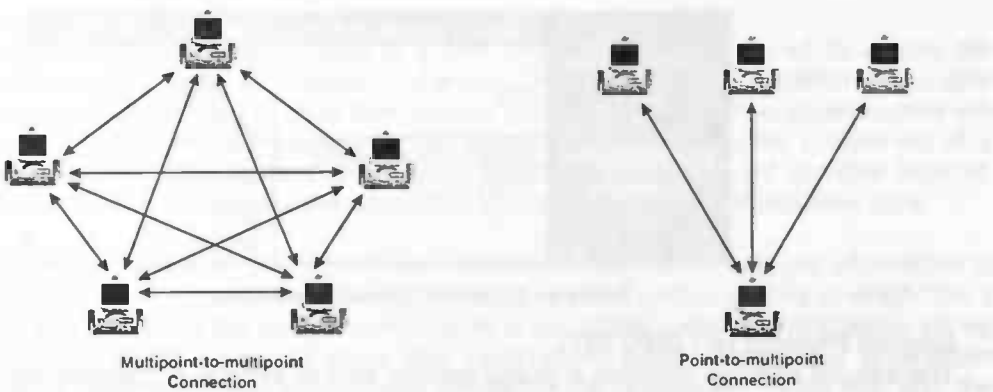


Figure 3-4: Types of multiparty connections

A problem is that the capacity of today's computer systems and networks is usually not sufficient to realise multiparty connections. The number of connections which can be established depends on variables like the capacity of the network to transport the multimedia streams and capabilities of the network to support for example multicasting, the capacity of the hardware to process these streams, the number of connections requested, the QoS requested, etc. In point-to-multipoint connections the used resources can be reduced by techniques such as multicasting, but in multipoint-to-multipoint connections, a large amount of network resources and processing capacity is needed.

For this reason, specialised Multipoint Control Units (MCUs) are often used in multipoint-to-multipoint configurations. An MCU is a specialised device which is capable of mixing audio and video signals. An MCU is mostly implemented in dedicated hardware, because of the computationally very intensive operations of decoding, mixing and coding video signals involved. However, software solutions are entering the market (e.g. White Pine's Multipoint, which is discussed in Section 4.5.3).

It is expected that this problem is solved when more processing capacity of computer systems and more capacity in computer networks becomes available, but with today's technologies performance of multipoint-to-multipoint multimedia communication is poor without using specialised hardware.

3.4 Transport and Control Protocols

The following paragraphs discuss a number of protocols are that suitable for, or are specially designed for the control and transport of multimedia data. Special attention is made here to the QoS management capabilities of the protocols discussed here.

Because these protocols mainly originate from the telecommunications area, they are usually designed using the protocol-centered paradigm (see Section 3.2). Protocols use services offered by other, lower level protocols, and offer services to higher level protocols or applications. In this way, a protocol stack of co-operating protocols is formed.

Transport and control protocols can be divided in two dimensions: connection-oriented and connection-less protocols, and circuit switched and packet switched protocols. Connection-oriented protocols first have to establish a connection between two endpoints, and use this connection to transmit data. Connection-less protocols can send data to the destination without first establishing a connection.

In a circuit switched network, a dedicated channel (or circuit) is established for the duration of the connection. In a packet switched network, each packet is sent individually over the network. IP is a packet switched protocol, and TCP is a connection-oriented protocol, so the Internet, which uses the TCP and IP protocols, is essentially a packet switched, connection-oriented network. The following table shows the type of the different protocols discussed here (the telephone network is not discussed in this section but is added as an example):

	circuit switched	packet switched
connection-less	n/a	IP IP-multicast UDP
connection-oriented	telephone network (PSTN)	ATM TCP RTP RSVP

Table 4: Different Connection-types

Internet Protocol (IP) [RFC 791]
The Internet Protocol provides a 'postal system' kind of service. It specifies the format of packets, also called datagrams, and the addressing scheme. Most networks combine IP with a higher-level protocol like the Transport Control Protocol (TCP) which facilitates reliable communication, or the User Datagram Protocol (UDP), which facilitates unreliable communication. IP is a connection-less protocol.

A new version of IP (IPv6) is currently being developed. This new protocol provides a much larger address space (the currently available amount of IP-addresses is running out due to the enormous growth of the Internet), and facilities for QoS reservation, by adding RSVP (see below).

Transport Control Protocol (TCP) [RFC 793]
The Transport Control Protocol enables two hosts to establish a connection and exchange streams of data. TCP runs on top of IP, and guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. The delivery of each packet is acknowledged by the receiver, and packets are retransmitted when needed. This makes TCP unsuitable for multimedia communications, because this

process can cause large delays in the delivery. Unlike IP, TCP is a connection-oriented protocol.

Disadvantages of TCP is the need for large routing tables in the routers, and routing can only be carried out in software, which is an order slower than implementation in hardware). Each incoming packet has to be decoded, interpreted by software, encoded and sent again, which takes a relatively large amount of time (compared with for example ATM, where routing can be implemented in hardware).

User Datagram Protocol (UDP) [RFC 768]
The User Datagram Protocol is a connection-less protocol, which runs on top of IP. UDP was developed to make transport of streaming data possible on the Internet. In contrast with the Transport Control Protocol (TCP), which offers a reliable service, UDP gives no guarantee whether a packet is received at the destination, or if it is received in the right order.

Real-Time Protocol (RTP) [RFC 1889]
The Real-Time Protocol is an Internet protocol for transmitting real-time data such as audio and video. It is primarily designed to satisfy the needs of multi-participant multimedia conferences. RTP itself does not guarantee real-time delivery of data, but it does provide mechanisms for the sending and receiving applications to support streaming data. Typically, RTP runs on top of the UDP protocol, although the specification is generic enough to support other transport protocols. A separate control protocol, RTCP (Real-Time Control Protocol), is used to monitor the quality of service and to convey information about the participants in an on-going session

RTP was released in 1996, and has received wide industry support since then. Netscape intends to base its LiveMedia technology on RTP, and Microsoft claims that its NetMeeting product supports RTP. RTP is also used by the ITU-T H.323 videoconferencing standard.

Resource Reservation Protocol (RSVP) [RFC 2205]
The Resource Reservation Protocol is a new protocol being developed to enable the Internet to support specified Qualities-of-Service. Using RSVP, an application will be able to reserve resources along a route from source to destination. RSVP-enabled routers will then schedule and prioritise packets to fulfil the QoS. RSVP is a chief component of a new type of Internet being developed, know broadly as an *integrated services Internet*. The general idea is to enhance the Internet to support transmission of real-time data.

In RSVP, reservations are made for 'flows', which are identified by address information in the IP-header. During data transfer, a router that receives a packet checks to which flow it belongs and schedules the packet transmission in accordance with the reservation set-up for that flow. RSVP uses soft state flow reservation, which means that reservation information must be update periodically, otherwise the reservation 'times out' and the allocated resources are released.

Reservations are made in a receiver-oriented style. Senders advertise information about flows in a Path message sent to all potential receivers. An end system interested in that flow generates a reservation message (containing a flow specification with information about the desired QoS), which travels towards the sender along the reverse path of the Path message. In this way, every receiver decides by itself how large a reservation it needs based on its own characteristics and requirements. This can lead to heterogeneous reservations from independent receivers.

Despite the advantages (provision of QoS in an IP-based network) of RSVP, it has some large disadvantages as well. The two largest disadvantages are that all components in the Internet have to be adapted to support RSVP, and that RSVP causes a relatively large overhead which may cost a significant amount of bandwidth, especially on large networks like the Internet. For more detailed information about RSVP and QoS management in the Internet, see [White, 1997] or [Ferguson, 1998].

Real-Time Streaming Protocol (RTSP) [RFC 2326]

The Real Time Streaming Protocol, or RTSP for short, is a proposed standard for controlling streaming data over the World Wide Web. RTSP grew out of work done by Columbia University, Netscape and RealNetworks and has been submitted to the IETF for standardisation. RTSP is designed to efficiently broadcast audio-visual data to large groups over IP networks. It is designed to work with established protocols such as RTP and HTTP to provide a complete solution for streaming media over the Internet.

IP-Multicast [RFC 1112]

IP-Multicast is an addition to the IP-protocol, and is used by applications to send data to the address of a multicast group, thereby sending the data to all receivers in that group. Without IP-multicast, the information would have to be sent to each receiver separately, so a lot of bandwidth would have been wasted. When a user wants to receive the information, he or she can announce him/herself to the multicast group. Figure 3-5 shows this difference between IP-multicast and traditional IP.

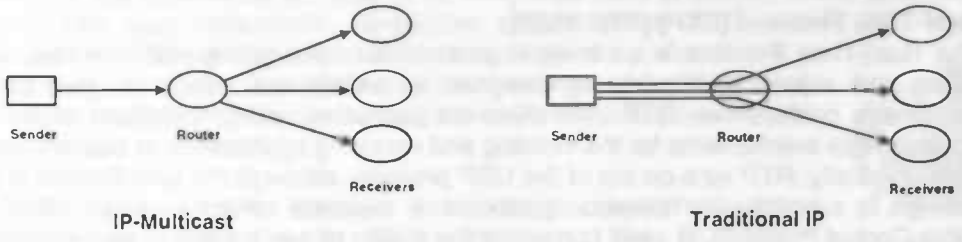


Figure 3-5: IP-Multicast vs. Traditional IP

When using traditional IP, a separate connection has to be set up between the sender and each receiver. When using IP-Multicast, the sender only needs to set up one connection with the multicast router, which sets up connections with each receiver. In Figure 3-5, this saves two connections.

Asynchronous Transfer Mode (ATM)

The Asynchronous Transfer Mode is a network technology based on transferring data in cells or packets of a fixed size. This fixed cell size makes that that packet switching can be implemented in hardware (instead of in software, like needed for IP), which is a large advantage over other protocols. This is the main reason for the high data transfer rates reached by ATM (current implementations of ATM support data transfer rates from 25 to 622 Mbps, compared with Ethernet which reaches speeds up to 100 Mbps).

Next to these high data transfer rates, another large advantage of ATM is that data, voice and video packets can be sent over the same connection simultaneously. The technology is also independent of the used physical network (like coax or optical fiber) so different types of networks can be connected using ATM. An ATM network can be used to make connections over very long distances. Together with the capabilities of transporting different kinds of data and the high bit-rates, ATM is currently being implemented by various telecommunication vendors.

ATM is a connection-oriented protocol; it creates virtual circuit (VC) between two endpoints for the duration of the connection. The type of service of a VC is fixed during a connection, in order to change the type of service, the current VC has to be broken down and a new VC has to be set-up. ATM provides four different types of service:

- *Constant Bit Rate (CBR)* - specifies a fixed bit rate so that data is sent in a steady stream. This is analogous to a leased line.
- *Variable Bit Rate (VBR)* - provides a specified throughput capacity but data is not sent evenly. This is a popular choice for voice and videoconferencing data.
- *Unspecified Bit Rate (UBR)* - does not guarantee any throughput levels. This is used for applications, such as file transfer, that can tolerate delays.
- *Available Bit Rate (ABR)* - provides a guaranteed minimum capacity but allows data to be burst at higher capacities when the network is free.

ATM provides multicast services by setting up a separate VC to each receiver, or by using a multicast server. In this case, the multicast server sets up a VC to each receiver. A large difference between QoS management in ATM and RSVP is that in ATM, resource management is hard and static (the type of service of a VC is fixed) and sender-initiated; in RSVP, resource management is dynamic and soft, and receiver-initiated.

It is expected that in the future ATM is used in back-bone networks for long-distance interconnections of LANs, while LANs will be either TCP/IP-based or ATM-based, depending on the services required and the investments already made in networking hardware and -software.

3.4.1 Protocol Stack of the Discussed Protocols

In this Section a protocol stack is designed of the protocols discussed in the previous section. The target of this protocol stack is to give a good insight in how the different protocols are related to each other.

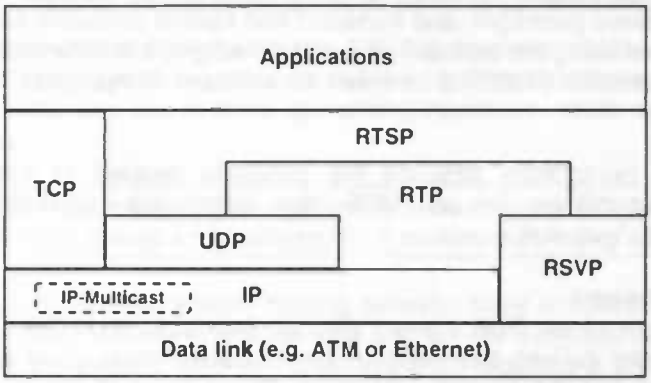


Figure 3-6: Transport and Control Protocols - Protocol Stack

IP-multicast is put inside IP in this Figure, because it is an optional addition to IP. Not all hardware and software components that support IP also support IP-multicast.

3.5 Requirements of Multimedia Systems in Open Distributed Environments

In the previous section requirements on hardware and protocols for multimedia systems were discussed, as well as design paradigms for (multimedia) systems. This section focuses on the additional issues and requirements posed on *distributed* multimedia systems, i.e. multimedia integrated in an open distributed environment, and describes how multimedia can be integrated in an ODE. The ODP Reference Model is used to provide the generic concepts needed for this integration of multimedia in open distributed environments (ODEs). An introduction to open distributed environments and the concepts of ODP-RM is given in Sections 2.4 and 2.5.

3.5.1 Integration of Multimedia in Open Distributed Environments

An advantage of ODEs with respect to existing solutions (e.g. Internet streaming solutions like RealAudio) is that ODEs open new possibilities for the set up, control and management of multimedia connections, and that they make existing control and management facilities better and easier to use.

These advantages can be realised due to ODE features like location transparency, failure transparency and platform interoperability. These facilities offer great opportunities to multimedia systems, like the possibility to allocate additional resources when needed, comprehensive support for the control and management of multimedia connections, QoS management and the possibility to establish connections between different types of devices (e.g. an audio connection between a telephone and a videoconferencing device).

Besides the advantages offered by an ODE to multimedia applications, an ODE benefits from the addition of multimedia capabilities, because in this way the capabilities of the ODE are considerably extended. These advantages make an ODE an important element in the integration of Information Technology and Communication Technology.

When integrating multimedia services in an ODE, the ODE must meet the issues discussed in the previous section. In most cases, this means that the ODE has to be adapted, to support for example protocols for the transport of multimedia streams. On the other hand, the multimedia services to be integrated have to comply to the properties of the ODE (the used meta-model, paradigms, etc.). Because ODEs usually tailor towards the object-centered paradigm, and transport and control protocols are usually designed using the protocol-centered paradigm, this difference may cause interaction problems between the different components in such a system.

Computational Viewpoint
(How to structure the system into functional objects ?)

The following paragraphs describe the concepts needed to add multimedia capabilities to an ODE, by using the ODP-RM *Computational Viewpoint*.

Multimedia Streams
The components of an ODE interact through *interfaces* (see Section 2.5.2). The ODP Reference Model defines two kinds of interfaces for interactions between objects: the signal interface and the operation interface. These types of interfaces are designed for discrete interactions, like operation requests. They are however not suitable for the continuous interactions (or streams) required by multimedia applications. Therefore an additional type of interface is needed, that supports continuous interactions. This type is called a *stream interface*, because of the streaming character of multimedia data:

- *Stream interface* - The stream interface describes behaviour, which consists of a single, non-atomic action that persists throughout the lifetime of the interface. A stream interface may consist of a number of unidirectional flows, each represented by a flow interface. It can be characterised as time-based (isochronous) information such as audio or video. A flow is an abstraction of a continuous sequence of data transmitted between interfaces. The stream interface signature contains the type of the flows and an indication of the causalities of the flows.

The addition of the stream interface concept to the ODP Reference Model is described in detail in [Leydekkers, 1997].

3.6 The Multimedia Binding Object

The ODP Reference Model uses the concept of object orientation to model distributed systems. Using this concept, the integration of multimedia and ODEs is accomplished by adding an object which facilitate the establishment of a multimedia connection (or *stream* connection) between two or more other objects, and by adding *stream interfaces*. This object is usually called the *multimedia binding object*. A definition of a multimedia binding object is given below:

> **multimedia binding object:** *a component which has the capability to set-up and control a multimedia connection (multimedia binding) between two or more endpoints, and offers an operational interface to control this binding*

The term 'binding' is defined in the ODP Reference Model, and refers to 'establishing a network path between'. A binding object abstracts from end-to-end connections and is responsible for compatibility checks between the involved interfaces.

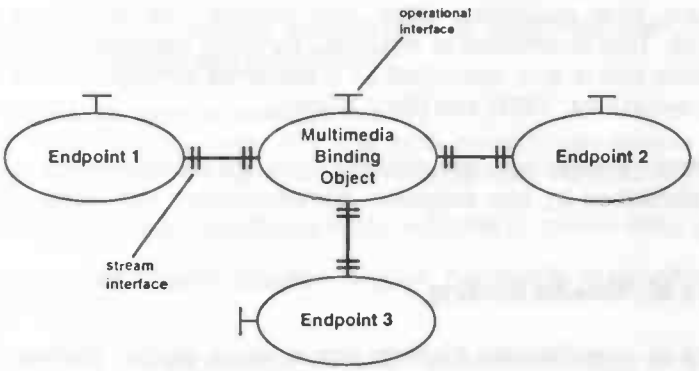


Figure 3-7: Multimedia Stream Binding viewed from the Computational Viewpoint

A multimedia binding object viewed from the ODP Computational Viewpoint has two or more stream interfaces, which are used to make a *multimedia stream binding* between two or more endpoints, and one or more operational interfaces, which are used to control the stream binding.

> **multimedia stream binding:** *a multimedia stream connection between two or more endpoints, set-up and managed by a multimedia binding object*

Figure 3-7 shows a multimedia stream binding between three endpoints. The multimedia binding object has three stream interfaces, and one control interface. An arbitrary object may be controlling the stream. One of the parties then requests a connection, possibly with a specified quality of service, to the binding object to establish the requested connection, with the quality of service requested.

A multimedia binding object represents *explicit bindings*, because separate control interfaces can be used to monitor and change the properties of the binding after the binding establishment. This capability makes it able to control and manage the binding externally, for example to change the QoS when needed, to store the used resources, to bill the user afterwards for usage of these resources, or add/remove endpoints from the binding.

The functionality of a multimedia binding object can be divided into six phases:

1. *set-up of a signalling channel* - An implicit binding is set-up between the binding object and the endpoints involved in the binding, for configuration purposes
2. *configuration negotiation* - Configuration and QoS data is passed out between the endpoints and the binding object. The binding object determines a configuration which is compatible with all endpoints and which meets the requested QoS. This configuration is eventually negotiated with the endpoints.
3. *connection set-up phase* - The binding object sets up a connection which is used for the actual binding.
4. *connection phase* - A communication path is established between the endpoints and multimedia data is transmitted (is 'streaming') over this connection
5. *configuration renegotiation phase* - The binding object tries to change the used configuration and/or QoS
6. *disconnection phase* - The binding object disconnects the binding between the endpoints by disconnecting all flows, and by releasing all resources

Phase 4 is always carried out by a protocol-centered standard, because of the performance advantages. The design paradigm used for the other phases depends on the standard used to implement the binding object.

In most object-centered implementations the multimedia binding object consists of one or more centralised control objects, and one or more objects residing at the endpoints. The objects at the endpoints mainly provide functions like abstraction from hardware devices, and expose interfaces to the centralised control objects and possibly to the objects residing in the peer endpoints. They also provide flow interfaces for incoming and outgoing flows. This distribution is observed by most standards that provide multimedia binding facilities and is also described as a preferred approach in the literature (see e.g. Chapter 4, [Leydekkers, 1997] and [Blair, 1998]).

In most protocol-centered implementations there are no centralised control entities, and a binding is established by two endpoints communication with each other via PDUs (see Section 3.2).

3.6.1 Controlling a Multimedia Binding

When setting up a multimedia binding, one or more parties involved in the binding are responsible for fulfilling tasks to accomplish this binding set-up. Responsibilities can be responsibility to establish (part of) the connection, configuration negotiation and configuration renegotiation. Different approaches to which object carry such responsibilities can be made. One approach is to use a centralised party which sets up and controls all aspects of the binding (e.g. determination of the used configuration, reservation of the needed resources, etc.). This approach is called *central responsibility*.

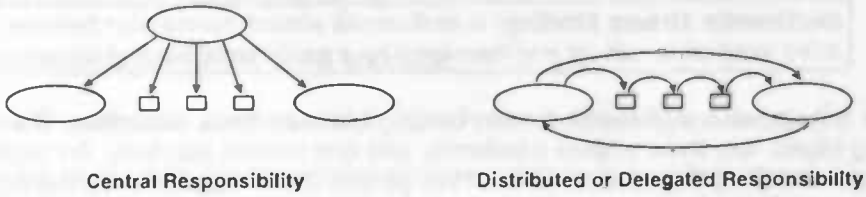


Figure 3-8 - Responsibility of control

Another approach is to start binding set up from one of the endpoints, which communicates with the other endpoint(s) and which distributes or delegates the control to the other endpoint(s) or the first network element used in the binding when necessary. No centralised control is available. This approach is called *distributed* or *delegated responsibility*. A good example of a distributed responsibility standard is the IP-standard.

3.6.2 Requirements of a Multimedia Binding Object Solution

In this section the requirements on a multimedia stream binding solution (an ODE which has multimedia capabilities) are discussed. This results in an extensive checklist, addressing requirements a multimedia stream binding solution should meet.

The checklist is divided into five sections: support for different protocols and codecs, support for quality of service management, support for multiple flow connections, support for multiparty connections, and support for stream synchronisation. Most of the items from the checklist are taken from [Leydekkers, 1997] and [Blair, 1998].

1. Support for different streaming protocols and codecs
The support for different protocols and codecs is necessary to make it possible to bind multimedia interfaces which use different streaming protocols and codecs. In theory, an interface should accept any relevant incoming flow, and should be able to bind any other relevant interface (the term 'relevant' is used here because it is not relevant to try to bind a stream interface to an operation interface).

The multimedia binding object should also be possible to bind interfaces which do not support all outgoing flows. It should for example be possible to bind a stream interface which supports audio and video flows to another interface which only supports audio flows.

The following items are added to the checklist:

1.1 Does the multimedia stream solution support all popular streaming protocols and standards?

1.2 Is the multimedia stream binding solution able to determine compatible interfaces through capability exchange and negotiation?

2. Support for Quality of Service management
In Section 3.3.3 QoS management of multimedia stream connections in ODEs was discussed. A multimedia stream binding solution should meet the following QoS issues:

2.1 Does the multimedia stream solution provide methods to control delay jitter?

2.2 Does the multimedia stream solution support bandwidth reservation and/or other resource reservation?

2.3 Does the multimedia stream solution provide negotiation of quality of service?

2.4 Does the multimedia stream solution provide dynamic quality of service management, such as monitoring, maintenance, policing and renegotiation of the quality of service requested?

3. Support for multiple flow connections
A multimedia stream solution should be able to bind to other multimedia stream interfaces which are of a different type. It should also be possible to combine flows (of possibly different types, and possibly from different producers) to one composite flow (for example, in a multipoint-to-multipoint videoconferencing session each party receives all audio flows originating from all other parties. All these flows have to be combined to one composite audio flow, which is then played through a speaker device. See the Figure below, which illustrates this concept).

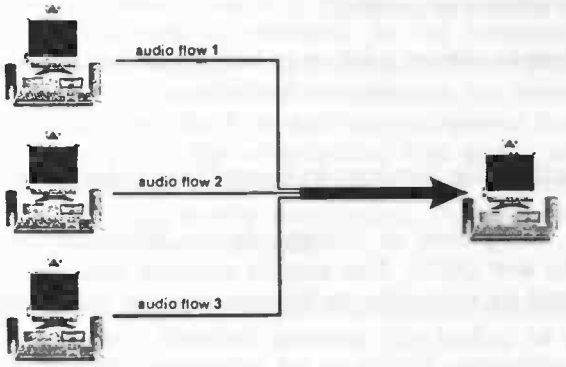


Figure 3-9: Combining audio flows to a single flow

The following items are added to the checklist:

3.1 Is the multimedia stream solution able to bind to multiple flows?

3.2 Is the multimedia stream solution able to combine flows (of possibly different types) to one composite flow?

4. Support for multiparty connections
Support for multiparty connections largely extend the capabilities of a multimedia stream binding solution. Many applications need point-to-multipoint or multipoint-to-multipoint connections (e.g. multiparty videoconferencing, video-on-demand). Multiparty connections can be added by using specialised hardware like a Multimedia Control Unit (see Figure 3-10), or by adding capabilities to the ODE to use other resources in the network to provide multiparty support (this possibility is briefly described in Section 3.6.3).

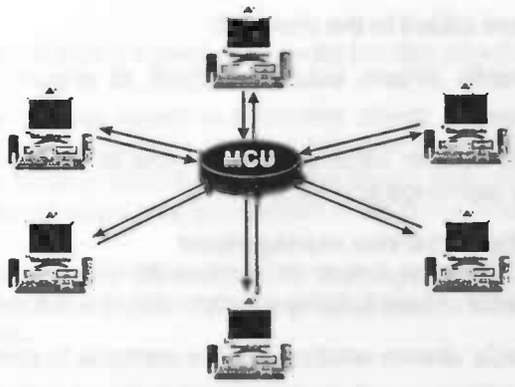


Figure 3-10: Video conferencing configuration using an MCU

The following items are added to the checklist:

4.1 Does the multimedia stream solution support point-to-multipoint connections?

4.2 Does the multimedia stream solution support multipoint-to-multipoint connections?

5. Support for stream synchronisation
Especially in video-conferencing applications, stream synchronisation is an important facility of a multimedia stream binding solution.

The following items are added to the checklist:

5.1 Does the multimedia stream solution support real-time flow synchronisation?

5.2 Does the multimedia stream solution provide methods to specify arbitrary synchronisation actions at run-time?

5.3 Does the multimedia stream solution provide synchronisation support for multiparty connections?

3.6.3 Advantages of Multimedia Services in Open Distributed Environments

As stated before, integration of multimedia in ODEs provides advantages for both multimedia systems and ODEs. This section explains these advantages and shows how they can be realised by using the multimedia binding object concept developed in the previous section.

Most of the services described here can also be realised without using an ODE. However, the facilities provided by an ODE makes it easier to provide these services, and mostly, better services can be provided.

Support for Connection Establishment
One of the properties of a multimedia binding object is that it is capable of binding possibly very different interfaces. When setting up a connection, the binding object can interrogate the endpoints about their capabilities using the ODE, and select an appropriate configuration for the binding (this depends on the type of connection, the requested QoS, etc.). When no compatible set of interfaces can be found, the binding object can check the ODE whether a resource is available which is able to convert one of the interfaces involved so a connection can still be established. Because endpoints and resources can be approached as components in the ODE, the binding object can communicate with them, transparently of the hardware and operating system these components are running on. This greatly eases the gathering of information needed for connection establishment.

Support for Quality of Service Management

Open distributed environments offer outstanding opportunities for Quality of Service management. By providing facilities for location transparency, replication and relocation, etc. parts of a multimedia service can be transferred to specialised systems. When the system load reaches an unacceptable level, the ODE can replicate parts of the service and relocates them on a second system, thereby doubling the processing capacity of the system.

Support for Multiparty Connections

Multiparty connections usually need very powerful resources to process all data flows. In a multipoint-to-multipoint videoconferencing session, a party has to receive and process the audio and video streams originating from all other parties.

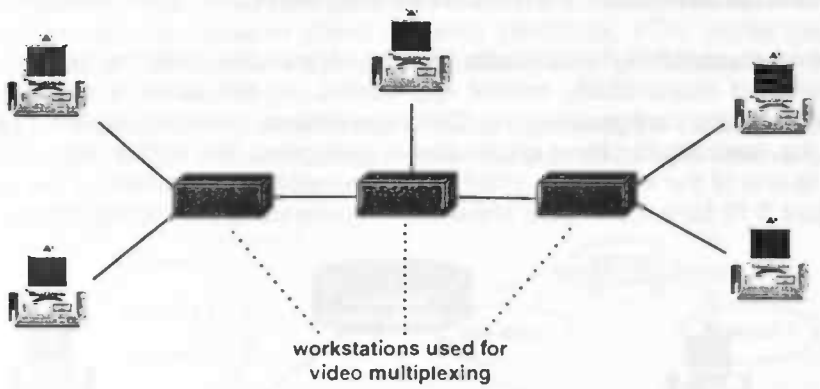


Figure 3-11: Multipoint-to-multipoint set-up in an ODE

Because of the capabilities for the control and management of streams, and the capabilities for increasing processing capacity by replication and relocation, an ODE is very suitable to support multiparty connections. In the videoconferencing example, multiple workstations could co-operate in multiplexing the audio and video streams (see Figure 3-11). In this set-up, three specialised workstations are allocated by the ODE to provide video multiplexing facilities. Each workstation multiplexes three video signals, so the workload is divided equally over the workstations. This set-up allows a multipoint-to-multipoint conference between five endpoints. When more endpoints want to join the conference, or when a higher video quality is requested, the ODE can allocate additional multiplexing resources to provide these facilities.

In this case, ODE facilities like migration and/or replication of components which perform multiplexing, load balancing and directory services (for lookup of available resources) make it much easier to manage resources for multipoint connections than conventional (protocol-centered) solutions.

Support for Flow Synchronisation

Flow synchronisation is needed when time-dependencies between different flows exist. A well-known example is the sound-lip synchronisation in a video connection. Because a multimedia binding object controls all flows in a connection, it can be used to co-ordinate flow synchronisation. A specialised synchronisation object, which is controlled by the binding object, can be used to provide synchronisation capabilities. For a discussion on flow synchronisation and the synchronisation object, see [Leydekkers, 1997], pp. 181-199.

3.7 The 'Hybrid' Multimedia Binding Object

In the previous section the properties and requirements of a multimedia binding object were treated. This section focuses on the methods to design a 'hybrid' multimedia binding object, as mentioned first by [Leydekkers, 1997]. The lexical meaning of the term 'hybrid' is 'crossover product', so a hybrid multimedia binding object is a combination of existing multimedia binding solutions or standards. Different combinations of binding objects can be made, giving different meaning to the term 'hybrid'. This indicates that this term is too general and should be defined more specifically for each combination.

The two combinations to design a hybrid multimedia binding object are:

- a hybrid multimedia binding object is a binding object realised by a standard which combines both central responsibility and distributed or delegated responsibility.
- a hybrid multimedia binding object is a binding object that combines object-centered and/or protocol-centered standards to establish multimedia bindings.

To make a distinction between these combinations, it is proposed to name the first *hybrid responsibility*, and to name the second *hybrid standards*. Both are discussed in the remainder of this Section.

3.7.1 The 'hybrid responsibility' multimedia binding object, or: who is in charge?

The 'hybrid responsibility' multimedia binding object uses both the central responsibility and distributed responsibility control approaches. In this case, a centralised object is responsible for the configuration and QoS negotiations. When all necessary information is known, the centralised object distributes or delegates the further establishment of the binding to one of the endpoints, which are responsible for establishing the actual binding. See Figure 3-12 for a multimedia stream connection using this 'hybrid control' approach.

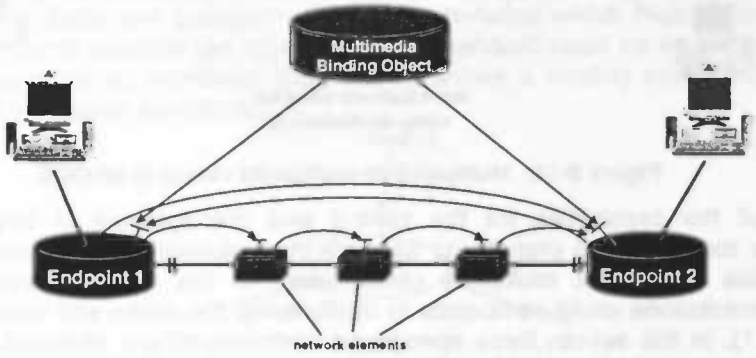


Figure 3-12: Multimedia Stream Connection using the 'Hybrid Responsibility' Approach

The hybrid responsibility approach integrates the 'best of two worlds'. It facilitates centralised responsibility for high-level binding set-up and configuration negotiation, so a high grade of (external) control is possible, but it also has the ability to delegate control to other standards to establish the actual connection. This makes it possible to use a wide variety of standards for connection establishment. Also, new standards can be easily used.

3.7.2 The 'hybrid standards' multimedia binding object, or: how to exchange information?

The other approach is to look at how different standards (which can be either object-centered or protocol-centered) can co-operate. This poses issues like for example how an object-oriented standard communicates with a protocol-centered standard, when such standards have to co-operate. Two ways of co-operation can be identified (see also Figure 3-13):

- *Federation* - multiple standards co-exist, possibly in separate address spaces. Co-operation between standards takes place by means of interface interactions, which can be object-interactions between object-centered standards, PDU exchanges between protocol-centered standards, or by using a *gateway* which translates between object-interactions and PDU exchanges.
- *Integration* - the implementations of two or more standards are combined, resulting in a new standard, which is able to act as each of the standards from which it is built of.

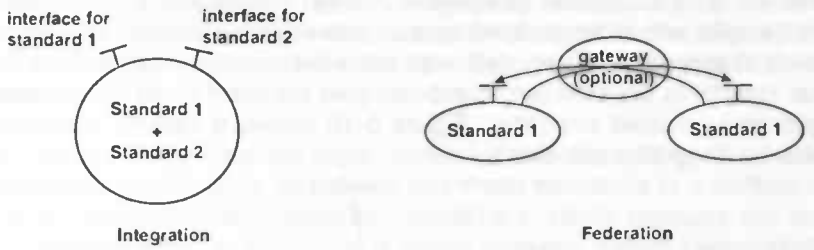


Figure 3-13: Integration vs. Federation

Federation

When two or more standards co-operate through federation, they have to communicate with each other in some way. When both standards are designed using the object-centered paradigm, they communicate through object interactions. When both standards are designed using the protocol-centered design, they communicate through message interfaces by exchanging PDUs (see Section 3.2). When the standards involved are designed using different paradigm a *gateway* is needed to perform the conversion between requests through object-interactions and requests through PDUs. Figure 3-14 shows these possible configurations.

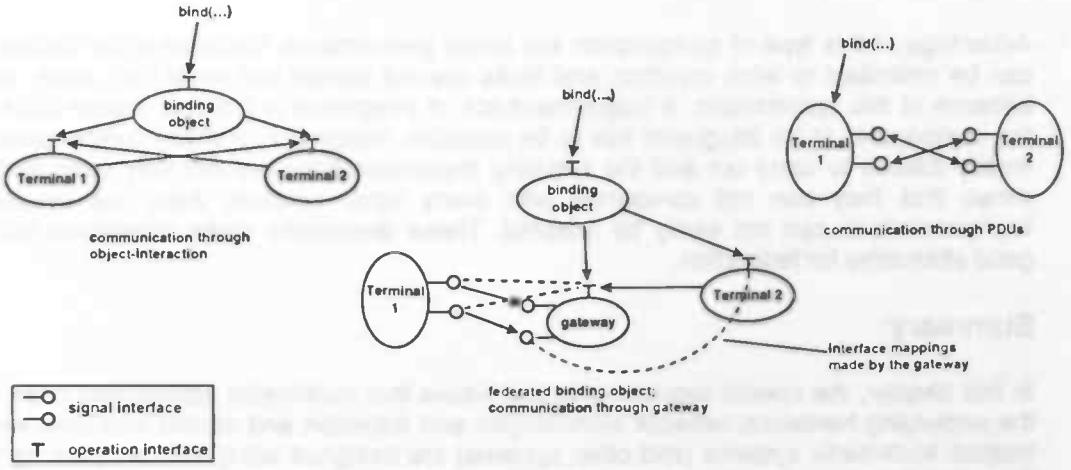


Figure 3-14: Possible federation configurations between standards

A gateway should be automatically utilised by the binding object when it is needed. Various gateways can exist in an ODE, providing conversions for specific federations. Another possibility is to design gateways with one specialised interface and one generalised interface. In this way gateways communicate with each other through the

generalised interfaces, so every conversion can be made by two co-operating gateways (see also Figure 3-15). This topic is however not investigated further in this thesis and is left open for further research.

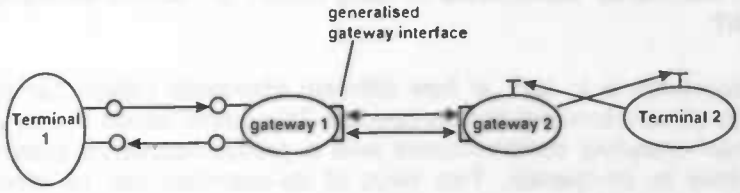


Figure 3-15: Using gateways with a generalised interface

The advantage of federation is relatively easy co-operation between standards without having to adapt these standards. A drawback of this approach is that tasks (like QoS negotiation) can be carried out more than once, because they are implemented differently by the standards involved. Prevention of these situation requires careful mapping of operations and services by the gateways, and detailed information about the behaviour of the standards.

Integration

When two or more standards co-operate through integration, their implementations are integrated. Unlike with federation, no special gateways are needed; one (or more) of these (integrated) standards are provided with extra interfaces which makes communication with other standards possible (an object-centered standard could for example be provided with a protocol-centered interface). Figure 3-16 shows a binding configuration with two terminals with integrated standards.

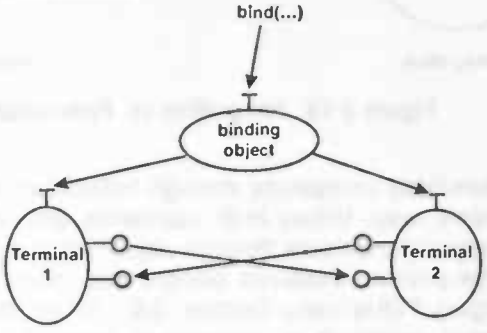


Figure 3-16: Using integrated standards in a binding

The terminals posses both an object- and a service-interface. Necessary conversions are made inside the terminals.

Advantage of this type of co-operation are better performance, because all components can be optimised to work together, and tasks are not carried out more than once, also because of this optimisation. A major drawback of integration is that the source code of the components to be integrated has to be available. Integration of these components is mostly difficult to carry out and the resulting implementations are not very open in the sense that they can not co-operate with many other solution, Also, the resulting implementations can not easily be adapted. These drawbacks make integration not a good alternative for federation.

3.8 Summary

In this chapter, the special requirements and issues that multimedia applications pose on the underlying hardware, network technologies and transport and control protocols were treated. Multimedia systems (and other systems) are designed using different paradigms. Two paradigms are introduced: the object-centered paradigm, where systems consist of objects with interfaces, communicating through object-requests, and the protocol-centered paradigm, where systems consist of a protocol stack of services. Each service consists of protocol entities communicating through Protocol Data Units.

Multimedia applications need powerful hardware, a computer network with large bandwidth capacity and support for quality of service management and multiparty connections. Today's computer networks, like LANs and the Internet, do not provide these facilities. To overcome these problems, specialised transport and control protocols are used to add multiparty and QoS support to these networks.

Also, the special issues that arise when multimedia services are integrated in an ODE are discussed. ODEs provide several facilities of which a multimedia service could benefit, like location transparency, replication and relocation, etc. On the other hand, ODEs also benefit of the addition of multimedia capacity, because the functionality of an ODE is extended by adding multimedia services.

The ODP Reference Model is used to describe how multimedia support can be added to an ODE. For this, the concept of a binding object, which is used to establish connection between objects is extended to a 'multimedia binding object', which is used to establish multimedia connections between objects. Also, the 'stream interface' concept has to be added, to provide interaction means for multimedia connections.

The multimedia binding object can usually be composed into a stream control object which provides the set-up, control and management functionality, a stream endpoint object, which abstracts from a stream endpoint and which provides configuration and QoS negotiation functions, and a multimedia device object, which abstracts from multimedia hardware devices.

A distinction can be made between object-centered and protocol-centered standards to establish multimedia connections. In order to use these standards in a multimedia binding object, a 'hybrid' approach is suggested by [Leydekkers, 1997]. Because this term can be explained in different ways, a distinction is made between 'hybrid responsibility', where both central responsibility and distributed responsibility is used to establish a binding, and 'hybrid standards', where a binding is established between heterogeneous endpoints, supporting both object-centered and protocol-centered standards.

Multiple standards can co-operate through federation or integration. When object-centered and protocol-centered standards are federated, a gateway is often needed to provide conversions between interfaces. When they are integrated, conversion is accomplished inside the integrated standards. Federation is preferred above integration, because in the case of integration the implementations of the standards used have to be combined, while when using federation the standards can stay separate and communicate through interfaces.

4 Solutions and Standards for Distributed Multimedia Systems

4.1 Introduction

This chapter treats a number of solutions and standards for the realisation of a distributed multimedia system. No distinction is made between vendor solutions, de facto standards and de jure standards, because sometimes a vendor solution is accepted as a de facto standard or implements de jure standards. The main difference is that vendor solutions often are commercial products, while de jure standards are provided by standardisation bodies. De facto standards often rise from vendor solutions and are sometimes 'promoted' to de jure standards.

The target of this survey is to give an overview of available solutions and standards for distributed multimedia systems. A number of these solutions and standards is selected to be analysed and compared in Chapter 5. To select solutions and standards that are of interest for this survey, the following criteria were used:

- The solution or standard should be based on an open distributed environment, or it should be easy to integrate it in an open distributed environment
- The solution or standard should be widely used, or it is expected that it will be widely used in the near future
- Both protocol-centered and object-centered standards should be selected
- Standards should be selected facilitating central responsibility and distributed responsibility

First, a survey of available solutions and standards was made. The resulting solutions and standards are summarised in Table 5.

Standard/Solution	Organisation	Short description
Control and Management of Audio/Video Streams	OMG (Object Management Group)	The OMG Control and Management of Audio/Video Streams model proposes a set of interfaces which implement a distributed media streaming framework, with much attention to the control and management aspects. The standard proposes interfaces in CORBA-IDL.
Telecommunications Information Networking Architecture - Network Resource Architecture	TINA-Consortium	The goal of the TINA-consortium is to define and validate a software architecture that will enable efficient introduction and management of new and sophisticated telecommunications services. The Network Resource Architecture (NRA) defines a set of generic concepts, which describe transport networks in a technology independent manner and provides the mechanisms for the establishment, modification and release of network connections.
H.32*x* series (H.320, H.321, H.322, H.323, H.324)	ITU-T	The H.32*x* series are ITU-T umbrella recommendations, which set standards for audio, video and data communications across different kinds of networks, like ISDN (H.320) or IP based networks (H.323). The standards address call control, multimedia management, codecs and multiparty connections.

Java Media Framework	SUN	The Java Media Framework (JMF) embeds multimedia content into Java applications and applets. It is designed to be protocol-neutral, content-neutral, with support for a wide range of file formats, popular streaming media protocols, and video on demand. It is planned to incorporate basic control and management functionality (like binding set-up and playback control functionality) in the JMF
DSM-CC (Digital Storage Media Command & Control), DMIF (Delivery Multimedia Integration Framework) and MPEG 4	DAVIC	These are solutions designed and/or used by Davic. They facilitate the transmission and control of audio and video streams over broadband and IP-networks. DSM-CC and DMIF are used for control and management issues, MPEG 4 is used for *content* coding.
Meetingpoint	White Pine	Meetingpoint is a software-based Multipoint Control Unit which is capable of establishing multipoint-to-multipoint H.323 videoconferencing connections.
Netmeeting	Microsoft	Netmeeting is a video-, audio- and data-conferencing tool that allows group communication and collaboration. Netmeeting implements the H.323 protocol for audio- and video-conferencing and the T.120 protocol for data-conferencing, and supports a large amount of video-conferencing hardware. A Developers Kit is also available.

Table 5: Solutions and standards for the realisation of a distributed multimedia system

Of these seven solutions and standards, the following are described in detail:

- OMG Control and Management of Audio/Video Streams
- TINA Network Resource Architecture
- ITU-T H.323

The other solutions/standards are discussed less extensive. The reason to choose these three solutions/standards is that the TINA Network Resource Architecture is representative for an object-centered standard and facilitates central responsibility, the ITU-T H.323 standard is representative for a protocol-centered standard and facilitates distributed responsibility, and the OMG Control and Management of A/V Streams standard incorporates features from both object-centered and a protocol-centered standards, and facilitates both central and distributed responsibility.

4.2 The OMG Control and Management of A/V Streams Specification

Because of the rising interest in multimedia applications, the OMG has passed out an RFP (Request For Proposal) on the control and management of A/V (audio/video) streams to add multimedia streams support to the CORBA-standard. The response of the RFP of IONA Technologies, in corporation with Lucent Technologies and Siemens Nixdorf A.G. was accepted by the OMG, and is recently added to the CORBA Telecoms domain technology document. This submission is discussed in the following sections.

The submission uses an architecture based on Figure 4-1. This Figure shows two stream endpoints, consisting of a stream interface control object, a flow data endpoint and a stream adapter, and a control and management object. Stream control operations are transmitted between the control and management object and the stream endpoints through an ORB Core, the flow is transmitted through a separate connection (this can for example be an ATM connection or an RTP connection over IP).

According to the control and management of A/V streams submission ([OMG A/V Streams]), the submission to the control and management of A/V streams "provides definitions of the components that make up a stream and for interface definitions onto Stream Control and management objects (see Figure 4-1, interface number 1a), and for interface definitions onto stream interface control objects (interface number 2b) associated with individual stream endpoints". Standard CORBA IDL is used to define the interfaces of the stream interface control objects.

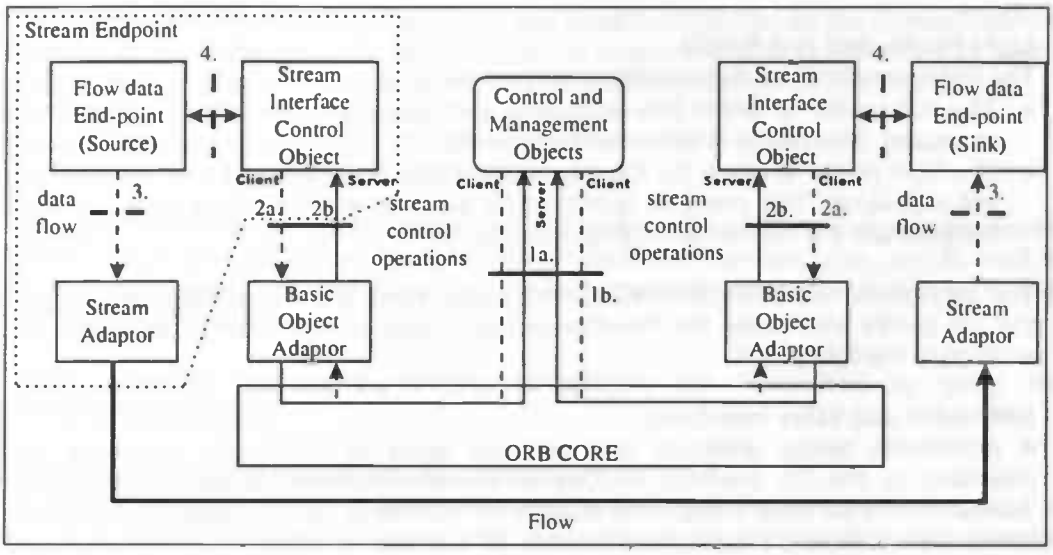


Figure 4-1: Example stream architecture from the RFP

4.2.1 Architecture

The solution as proposed by the submission consists of the following components:

- *Streams* - represented by the StreamCtrl interface
- *Multimedia devices and virtual multimedia devices* - represented by the MMDevice and VDev interfaces
- *Stream endpoints* - represented by the StreamEndPoint interface
- *Flows and flow endpoints* - represented by the FlowConnection and FlowEndPoint interfaces
- *Flow devices* - represented by the FDev interface

A *stream* represents continuous media transfer, usually between two or more *virtual multimedia devices. A stream endpoint* terminates a stream. A stream may contain multiple unidirectional *flows*, so a *flow endpoint* may be either a source (or producer) or sink (consumer). A *multimedia device* abstracts one or more items of multimedia hardware and acts as a factory for *virtual multimedia devices*. A multimedia device can support more than one stream simultaneously. Figure 4-2 shows a typical stream configuration using the OMG specification.

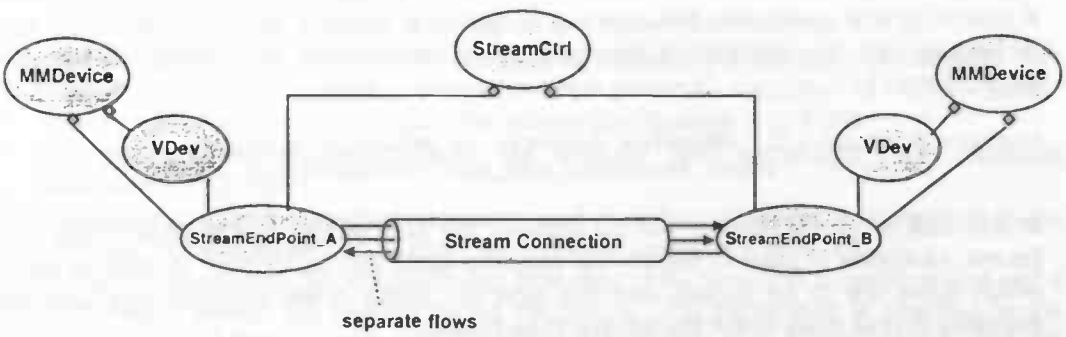


Figure 4-2: Stream Configuration using the OMG spec.

The StreamEndPoint interface type has two specialisations StreamEndPoint_A and StreamEndPoint_B. This distinction is made in order to help the implementation to distinguish between stream endpoints and to determine the relative direction of flows. For example, in a stream between an A- and a B-endpoint containing of two flows video1 and video2, if video1 is produced by the A-endpoint it has to be consumed by the B-endpoint, and if video2 is produced by the B-endpoint, it has to be consumed by the A-endpoint. In this way, A and B parties act like a 'plug and socket'.

Light Profile and Full Profile

The OMG specification. distinguishes two 'profiles':

- The 'full profile' in which flow endpoints and flow connections have accessible IDL interfaces. This profile is optimised for flexibility
- The 'light profile' in which the IDL interfaces of flow endpoints and flow connections are not accessible. This profile is optimised for systems which need to minimise memory footprint and the number of CORBA invocations.

The light profile uses the StreamCtrl, MMDevice, VDev and StreamEndPoint interfaces, the full profile also uses the FlowConnection, FlowEndPoint and FDev interfaces, in addition to the light profile.

MMDevice and VDev Interfaces

A multimedia device abstracts one or more items of multimedia hardware and is described by the IDL interface MMDevice. A multimedia device can be connected or 'bound' to one or more compatible multimedia interfaces using a stream. A multimedia device can in theory support any number of streams to other multimedia devices, in practice, this number is limited by the available resources.

A multimedia device creates a virtual multimedia device (VDev) and a stream endpoint (StreamEndPoint) for each stream connection, representing the device specific and network specific aspects of a stream endpoint respectively.

Virtual multimedia devices have configuration parameters associated with them, describing the capabilities of the hardware device abstracted by the VDev. The VDev interfaces also have operations used for configuration and negotiation purposes. This negotiation/configuration procedure is initiated by the StreamCtrl interface, by calling the operation `set_peer()` on the 'A-party' VDev with the 'B-party' VDev as a parameter, and then calling `set_peer()` on the 'B-party' VDev with the 'A-party' VDev as a parameter. These calls cause the VDev interfaces to negotiate about the configuration to be used for the stream connection. This will incorporate selecting protocols and QoS parameters supported by both VDevs, and by finding compatible source and sink endpoints for each flow in the stream.

StreamCtrl Interface

The StreamCtrl interface abstracts a continuous media transfer between virtual devices. It supports operations to bind multimedia devices using a stream, as well as operations to start and stop a stream. The StreamCtrl interface can be extended to support more complex functionality (like spooling streams).

A stream or flow connection between two endpoints is called a 'binding' in the OMG spec. A binding can be established by the StreamCtrl interface by calling the operation `bind_devs()`:

```
boolean bind_devs(in MMDevice a_party, in MMDevice b_party,
                  inout streamQoS the_qos, flowSpec the_spec)
```

In this call, `a_party` and `b_party` address the MMDevice-objects which have to be bound, `the_qos` is used to specify the required QoS, and `flowSpec` is used to specify which flows are to be bound, and the type of those flows. A value `nilFlowSpec` indicates that all flows in the stream are to be bound.

StreamEndPoint interface

A stream endpoint logically contains and controls the flow endpoints for each of the individual flows in a stream. There are two specialisations of the StreamEndPoint interface: StreamEndPoint_A and StreamEndPoint_B. The reason to make this distinction is to help the application to determine the direction of a flow.

The actual stream is established between two stream endpoints, so the StreamEndPoint interface is responsible for establishing the actual stream connection. This done by calling the operation `connect()` on a StreamEndPoint_A or StreamEndPoint_B interface:

```
boolean connect(in StreamEndPoint responder, inout streamQoS the_qos,
                in flowSpec the_spec)
```

This operation is invoked by the StreamCtrl interface on one of the two stream endpoints, where `responder` addresses the peer StreamEndPoint interface, `the_qos` is used to specify the required QoS, and `flowSpec` is used to specify which flows are to be bound, and the type of those flows.

When using the full profile, stream connections are established by using the FlowEndPoint interface.

FlowConnection and FlowEndPoint Interfaces
The FlowConnection interface is the flow-level analogue of the StreamCtrl interface. A FlowConnection interface abstracts from a uni-directional continuous media transfer between two or more flow endpoints, where a StreamCtrl interface abstracts from a (possible bi-directional) continuous media transfer between two stream endpoints.

Just like the StreamEndPoint interface, there are also two specialisations available of the FlowEndPoint interface: the FlowProducer and the FlowConsumer interfaces. A flow is directed from a FlowProducer interface to a FlowConsumer interface. A FlowConsumer interface can be put in 'listening mode' by calling the operation `go-to_listen()` on it, the endpoint is then ready to receive a flow. A FlowConsumer endpoint can start transmitting the flow by calling the operation `connect_to_peer()` (in the point-to-point case) or `connect_mcast()` (in the point-to-multipoint case) on it. Furthermore, the FlowEndPoint interface supports the same operations as the VDev interface.

The FlowConnection interface supports operations to add and remove FlowProducer and FlowConsumer endpoints to a flow connection and is able to connect and disconnect two flow endpoints by calling the operations `connect()` and `disconnect()` on it.

FDev Interface
The FDev interface is the flow-level analogue of the MMDevice interface. It serves as a factory for FlowProducer and FlowConsumer interfaces by providing the operations `create_producer()`and `create_consumer()`. It also supports operations to create and destroy a binding with another FDev interface.

4.2.2 The Simple Flow Protocol (SFP)

The OMG spec defines its own flow protocol to facilitate the transport of streams. This protocol provides the three types of transports which should be supported by the framework:

- Connection-oriented transport: used for reliable data transfer (e.g. TCP)
- Datagram-oriented transport: used for efficient and lightweight, but unreliable datatransfer (e.g. UDP)
- Unreliable connection-oriented transport: used to deliver messages in sequence to the endpoint, but messages can be dropped or contain errors (e.g. ATM AAL5).

Furthermore, information like timestamping, indication of the source of a media packet and a synchronisation source are needed by the framework. This information is provided by the RTP protocol, but this protocol has the disadvantage that it is internet-based, and the framework should also support other network environments.

To support all the features discussed in the previous paragraph, a specialised protocol which works on top of various transport protocols and provides architecture independent flow content transfer. This protocol is called the Simple Flow Protocol (SFP). It is however not mandatory for an implementation to support SFP. When an implementation does not support SFP, other endpoints which do support SFP can switch this feature off.

4.2.3 Binding Establishment Using the Light Profile

In this section the establishment of a point-to-point stream binding using the light profile is explained. The establishment of a point-to-multipoint binding is explained in Section 4.2.5.

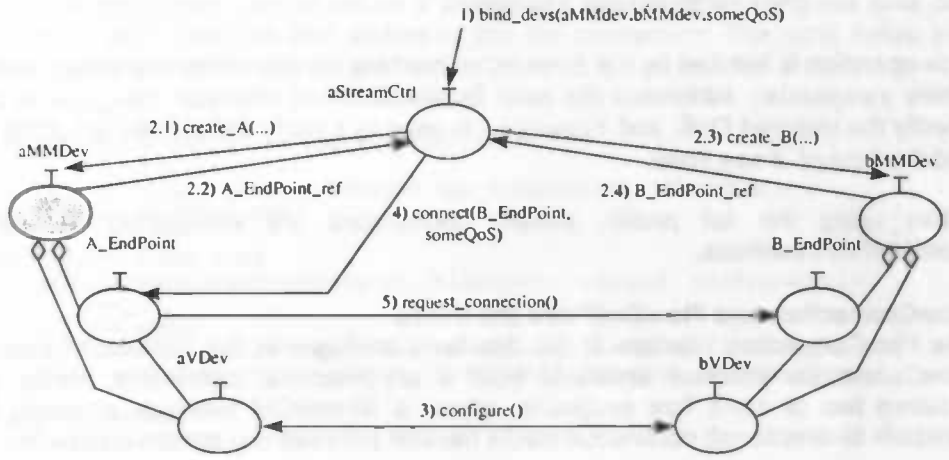


Figure 4-3: Establishing a point-to-point binding

Figure 4-3 shows the operation sequence of a point-to-point stream binding set-up using the light profile. The sequence is started by an application calling the operation `bind_devs()` on an instance of the StreamCtrl interface (1). This call holds the references to the MMDevices that are to be bound, the flows to be bound and the requested QoS.

The StreamCtrl issues a `create_A()` request on the MMDevice which is the calling party and a `create_B()` request on the MMDevice which is the called party (2.1, 2.3). These calls causes the MMDevices creating a StreamEndPoint_A and a VDev object on the calling side, and a StreamEndPoint_B object., and a VDev object on the called side. References to these objects are returned to the StreamCtrl (2.2, 2.4).

The next step is the StreamCtrl calling `set_peer()` on each VDev (3). This causes the VDevs to negotiate about the desired configuration to meet the requested QoS, and it is checked wether the flow endpoint are compatible so all flows in the stream can be bound. When the configuration is ready, the StreamCtrl calls the operation `connect()` on the StreamEndPoint of the calling party (4). This StreamEndPoint contacts the StreamEndPoint of the peer party to retrieve the connection address of this party, by calling the operation `request_connection()` on the peer party (5). When the request_connection() call has returned, the actual connection can be established.

4.2.4 Binding Establishment using the Full Profile

The establishment of a stream binding using the full profile is very much like binding establishment using the light profile, with some slight differences due to the use of additional interfaces that make it possible to control each flow in a stream separately. Figure 4-4 shows the operation sequence of a point-to-point stream binding set-up using the full profile.

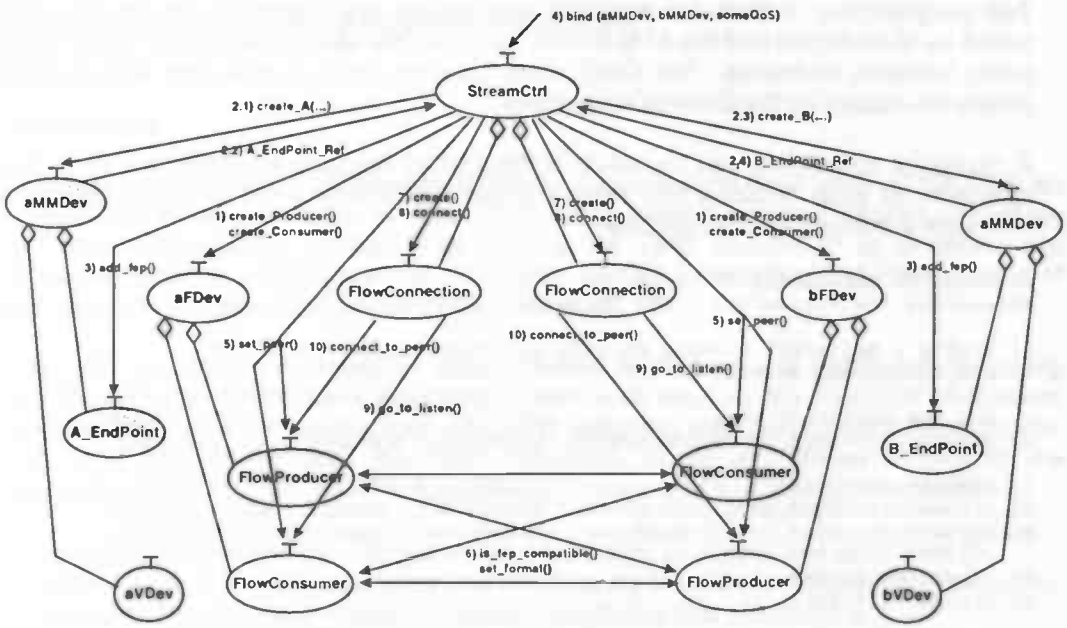


Figure 4-4: Establishing a point-to-point binding using the full profile

The main differences with the light profile are the addition of the FlowConnection, FlowEndPoint and FDev interfaces. The operation sequence is the following:

First, the application programmer creates the needed FlowProducer and FlowConsumer interfaces on each endpoint, by calling the operations `create_Producer()` and `create_Consumer()` on the FDev interfaces associated with each endpoint. (1). The application programmer also creates StreamEndPoint interfaces and VDevs by calling the operations `create_A()` and `create_B()` on the MMDevices of the endpoints (2.1 and 2.2).

These interfaces are 'connected' to the StreamEndPoint interfaces associated with each endpoint by calling the operation `add_fep(FEP)` on the StreamEndPoint interface, where `FEP` is the name of the FlowProducer or FlowConsumer interface (3).

The next step is the application programmer to call the operation `bind()` on the StreamCtrl interface, with the addresses of the StreamEndPoints to be bound as arguments. (4) This causes the StreamCtrl interface to first call `set_peer()` on each FlowEndPoint interface (5). The FlowEndPoint interfaces try to find compatible peer FlowEndPoints by calling the operations `is_fep_compatible()` and `set_format()` on the available peer FlowEndPoints (6). When these calls return successfull then each FlowProducer interface can be connected to a compatible FlowConsumer interface. The StreamCtrl now creates FlowConnection objects for each FlowProducer-FlowConsumer pair (7).

The StreamCtrl calls the operation `connect()` on each FlowConnection interface, with the addresses of the FlowEndPoints to be bound by that interface as parameters (8). The FlowConnection interfaces now call the operation `go_to_listen()` on the FlowConsumer interfaces, to set these interfaces ready to accept incoming flows (9). The last action is the FlowConnection interfaces calling the operation `connect_to_peer()` (10). This causes the FlowProducer interfaces to start transmitting flows to the associated FlowConsumer interfaces.

4.2.5 Multipoint Connections

The OMG spec supports point-to-multipoint (or multicast) connections, and according to the specification, multipoint-to-multipoint connections are easy to create by using the objects designed for point-to-multipoint connections as 'building blocks'.

Two programming models for multicast connections are identified: the internet-model, based on IP-multicast and the ATM model, based on the ability to establish Virtual Circuits (VCs) between endpoints. The OMG spec supports both models; the implementation details are hidden by the StreamCtrl interface.

A multipoint connection can be set-up a StreamCtrl interface by calling the bind_devs() operation for each endpoint that wishes to join the connection. The code below shows how such a connection is established.

```
// Create a StreamCtrl
StreamCtrl *my_StreamCtrl = new StreamCtrl();

// Create a multicast binding between the VideoSource, VideoSink1
// and VideoSink2 MMDevices:

// Add the multicast root
my_StreamCtrl->bind_devs(VideoSource, nilObject, someQoS, nilFlowSpec);

// Add the two sinks
my_StreamCtrl->bind_devs(VideoSource, VideoSink1, someQoS, nilFlowSpec);
my_StreamCtrl->bind_devs(VideoSource, VideoSink2, someQoS, nilFlowSpec);

// Start the stream
my_StreamCtrl->start();

// …Stop the stream and unbind
my_StreamCtrl->stop();
my_StreamCtrl->unbind();
```

First, a multicast root is set-up between a source and two sinks. After this, the sinks are bound to the source; the parameter `nilFlowSpec` denotes that all flows in the stream have to be bound. After the multicast connection is set-up, the stream is started by calling the operation `start()` on the StreamCtrl. Parties can be added and removed both before and after the stream has been started.

When a multipoint connection is set-up as described in the example above, the StreamCtrl interface creates a multipoint binding by using a specialised interface, the `MCastConfigIf` interface. This interface serves as a kind of multicast 'bridge' for the VDev interface of the multicast source by broadcasting configuration information to the sink endpoints. 'Behind the screens', the following operation sequence takes place when creating a multipoint connection:

1. The application programmer calls `my_StreamCtrl->bind_devs(VideoSource, nilObject, someQoS, nilFlowSpec)`.
2. The my_StreamCtrl object creates an instance of StreamEndPoint_A by calling `SEP_A = VideoSource->create_A(..., aVDev, ...)` and an instance `mc` of MCastConfigIf. It then calls `aVDev->setMCastPeer(mc, ...)`
3. The application programmer calls `my_StreamCtrl->bind_devs(VideoSource, VideoSink1, someQoS, nilFlowSpec)`
4. The my_StreamCtrl object creates an instance of StreamEndPoint_B by calling `SEP_B = VideoSource->create_B(..., bVDev, ...)`.
5. It then calls `mc->set_peer(bVDev, ...)` followed by `A_SEP->connect_leaf(B_SEP, ...)`.
6. If this call completes successfully then `B_SEP` is bound to a multicast tree with `A_SEP` being the root.

Setting up a multicast connection using the full profile works in a similar way. The FlowEndPoint interface has the same functionality as the VDev interface, so the MCastConfigIf interface can also be used as a multicast 'bridge' for the FlowEndPoint interface.

4.3 The TINA Network Resource Architecture

4.3.1 Introduction

The Telecommunications Information Networking Architecture Consortium (TINA-C) is an international initiative from the computer industry, telecommunication network operations and telecommunication vendors. The goal of the TINA consortium is to define and validate a software architecture that will enable efficient introduction and management of new and sophisticated telecommunications services.

The TINA architecture is based on object-oriented technology and distributed computing and incorporates results from international standards such as the ISO ODP Reference Model, OMG's CORBA architecture, and ATM switching and management technologies. The TINA architecture uses the ODP Reference Model and refines it towards the telecommunications area. The TINA architecture consists of four sub-architectures:

- The *Service architecture* - defines a set of concepts, principles and guidelines for constructing, deploying, operating and withdrawing of telecommunications information services. The service architecture defines the objects that constitute a telecommunication service, and how these objects should be used and interact.
- The Network *Resource architecture* - defines a set of generic concepts, which describe transport networks in a technology independent manner and provides the mechanisms for the establishment, modification and release of network connections.
- The *DPE architecture* - defines the information, computational and engineering concepts that are used in the other sub-architectures to specify telecommunication services. Additionally, the DPE architecture defines a generic DPE, which is used as the platform on which TINA compliant services are built.
- The *Management architecture* - provides generic management principles and concepts for the management of services, network and computing platform.

This section focuses on the TINA *Network Resource Architecture*. This part of the TINA architecture describes how to set-up, maintain and release telecommunication connections, as well as managing these resources in a TINA-C compliant architecture [TINA NRA, 1997]. The TINA Network Resource Architecture is based on the TINA Network model.

4.3.2 The TINA Network Model

A TINA network is a transport network (TN) that is capable of transporting multimedia information. This information is passed between endpoint through stream interfaces. A TINA network also uses a kernel transport network (KTN) to transport request and reply messages for computational objects via operational interfaces. This principle is called separation of call and connection control.

The TINA Network Model is described from a logical (application level) view, describing end-to-end connections, and a physical view, describing physical connections. In the logical view, stream interfaces are connected through *stream flows*, or Stream Flow Connections (SFCs). A stream flow is a unidirectional data stream connecting two Stream Flow Endpoints (SFEPs). An SFEP can be either a source or sink, but not both; an SFC can connect a source SFEP to one or more sink SFEPs. A *stream binding* is a concept that represents a collection of stream flows. See also Figure 4-5.

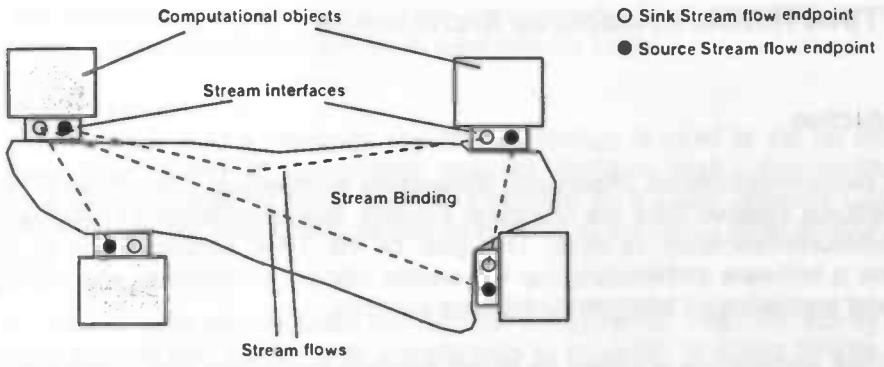


Figure 4-5: TINA Network (logical view)

In the physical view, a TINA network is divided into a Connectivity Layer Network (CLN), representing a transport network consisting of a heterogeneous collection of switching resources, transmission resources and adapters, and Customer Premises Equipment (CPE), representing the communication resources, like telephones or multimedia devices. A CPE can be attached to several connectivity layer networks.

A Connectivity Layer Network connects communication endpoints *called Network Flow Endpoints* (NFEPs). An NFEP can be either a source, sink, or both. NFEPs hold information about properties, QoS, etc. NFEPs are connected via *Network Flow Connections* (NFCs). These connections may be uni- or bi-directional. A stream flow is composed of one or more network flow connections and two or more *Terminal Flow Connections* (TFCs). The TFCs carry out necessary adaptations between SFEPs and NFEPs. See also Figure 4-6.

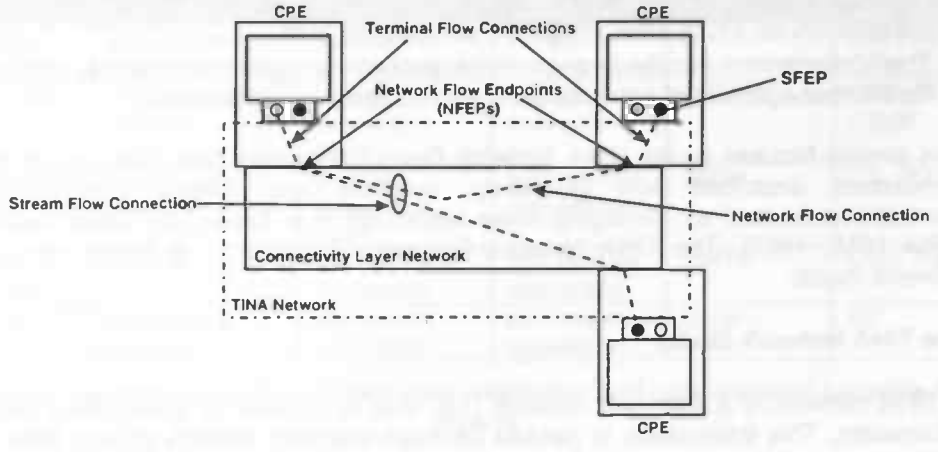


Figure 4-6: TINA Network (physical view)

A connectivity layer network is made up of one or more interconnected *layer networks*, which are networks based on a single technology transporting information of a single format. Examples of layer networks are ATM Virtual Path (VP) and Virtual Channel (VC) networks (see Section 3.4) and Frame Relay networks (not discussed here). Inside a layer network connections are modelled by *trails*, connecting *Network Trail Termination Points* (NWTTPs). A trail can be point-to-point uni-directional, point-to-point bi-directional or point-to-multipoint uni-directional. A layer network is decomposed into subnetworks that are interconnected by *topological links*, connecting *Network Connection Termination Points* (NWCTPs). An arbitrary segment of a trail can be represented by a *tandem connection*, which is an arbitrary series of contiguous subnetworks. Figure 4-7 shows these concepts.

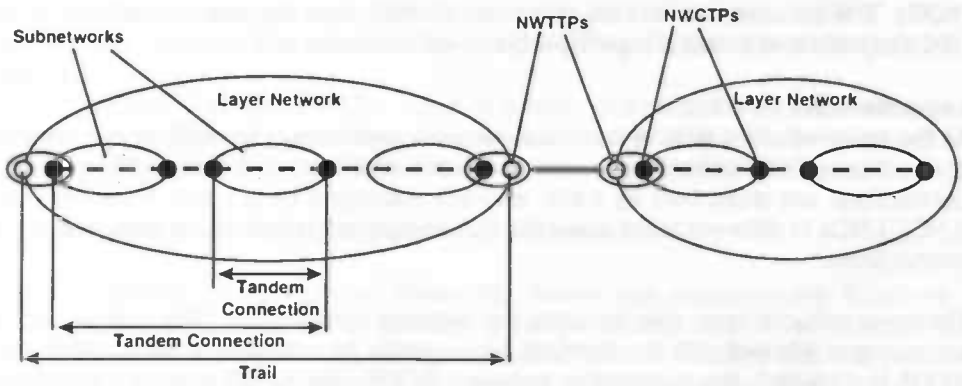


Figure 4-7: Layer Networks, Subnetworks, Trails and Tandem Connections

4.3.3 Connection Management

TINA Connection Management has two goals [TINA NRA, 1997]:

1. To specify reusable functions which manage network transport resources, independently of transmission and switching technology. The functions will be deployed as a set of distributed, interoperable software components.
2. To specify a generic network information model to support the management activity mentioned above.

The structure of the Connection Management Architecture is based on the TINA network modelling concepts, which are discussed in the previous Section. It is divided into a number of conceptual levels, that work together to transform service layer communication requirements into constructs that can be implemented In the resource layer. Each level has its own set of environmental and organisational concepts (e.g. 'session' and 'network' or 'subnetwork'), information models (e.g. service session graphs and connection graphs) and associated computational objects.

Service Level

The service level is not part of the NRA, but it uses the services of the communication level provided by the Connection Session Manager (CSM) to set up service-level connections. At this level, connections are described in terms of stream bindings between Stream Interfaces (SIs), modelled by information models like service session graphs (SSGs). The Service Session Manager (SSM) is responsible for setting up the overall stream binding. It translates the stream binding description into a set of stream flow connections, and uses a communication session to establish them.

Communication Level

A communication session is associated with a single client service session. The communication session has a terminal component (a Terminal Communication Session Manager, TCSM) and a network component (a Connection Session Manager, CSM). The communication session is responsible for end-to-end connections between Stream Flow Endpoints. These connections are described by Stream Flow Connections, which are modeled by a Logical Connection Graph.

The CSM uses the SFEP descriptions to determine the associated TCSM and Network Flow End Point (NFEP) or NFEPpools. The CSM contacts each TCSM associated with the SFC so that the terminal and network parts of the SFC can be connected. The CSM translates the SFCs to Network Flow Connections (NFCs) and uses the connectivity session to establish the NFCs.

Connectivity Level

Each connectivity session is associated with a client communication session. A connectivity session is controlled by a Connection Coordinator (CC), and is concerned with end-to-end network connections between NFEPs. A connectivity session presents a technology independent network view, which is modeled by a Physical Connection Graph

(PCG). The connectivity session components determine the layer network(s) to use, map NFCs to trails and locate a layer network to establish the trail.

Layer Network Level
At the layer network level a particular network technology is used to set up end-to-end connections, independent of connectivity provider domains and subnetworks. These connections are described by trails, and are managed by a Layer Network Coordinator (LNC). LNCs in different layer networks can co-operate to with each other through tandem connections.

The layer network level also includes the terminal components. The connectivity provider components interact with the terminal components to establish or select NFEPs. Once a NFEP is selected, the connection between SFEP and NFEP can be completed by the TCSM, which must be informed of the selection by the Terminal Layer Adapter (TLA) or the CSM.

Subnetwork Level
A subnetwork is associated with a particular network technology within a single domain. A Connection Performer (CP) offers management services to establish network connections. The CP manages the elements in a subnetwork (e.g. switches, routers) to establish and control these connections.

Table 6 summarises these levels, and the associated information and computational concepts.

Level	Environment Concept	Information Model	Connection Concept	Connection terminators	Associated Computational Objects
Service	service session	service session graph	stream binding	SI	UAP, SSM, USM, SSF
Communic ation	communication session	logical connection graph	stream flow connection	SFEP	CSM, TCSM, CSMF
Connection	connectivity session	physical connection graph	network flow connection	NFEP	CC, FCC, CCF
Layer network	network domain	trail, tandem connection	trail, tandem connection	NWTP	LNC, TM, TCM
Subnet-work	subnetwork, element	subnetwork connection	subnetwork connection	Edge	CP

Table 6: Conceptual levels and associated information and computational concepts and objects

4.3.4 TINA NRA from the ODP-RM Information Viewpoint

TINA uses connection graphs to describe connections from the ODP-RM Information Viewpoint. Different types of connection graphs are used to describe bindings from different conceptual levels. The following connection graphs are defined:

- *Service Session Graph (SSG)* - used to describe a session and the relationships between the elements (users, resources) of a session. SSGs are managed by SSM objects.
- *Logical Connection Graph (LCG)* - used to describe the concept of stream binding. A LCG consists of Stream Flow Connections (SFCs) connecting Stream Flow Endpoints (SFEPs). Connections exist between root SFEPs and leaf SFEPs; one root SFEP may be connected to several leaf SFEPs by using SFC-Branches. LCGs are managed by CSM objects.
- *Physical Connection Graph (PCG)* - used to specify end-to-end network connectivity. A PCG consists Network Flow Connections (NFCs) connecting Network Flow Endpoints (NFEPs). Connections exist between root NFEPs and leaf NFEPs; one root NFEP may

be connected to several leaf NFEPs by using NFC-Branches. NFEPs reference addressable Network Termination Points (NWTPs). PCGs are managed by CC objects.

- *Nodal Connection Graph (NCG)* - used to specify the connectivity requirements in a terminal. A NCG consists of one or more Terminal Flow Connections (TFCs), interconnecting SFEPS and NFEPs. NCGs are managed by TCSM objects.

4.3.5 TINA NRA from the ODP-RM Computational Viewpoint

From the ODP-RM Computational Viewpoint, connection management functions are described as provided by a set of interacting computational objects, of different types. Figure 4-8 gives an overview of the computational objects associated with the connection management architecture. It groups them according to conceptual levels introduced in Section 4.3.3.

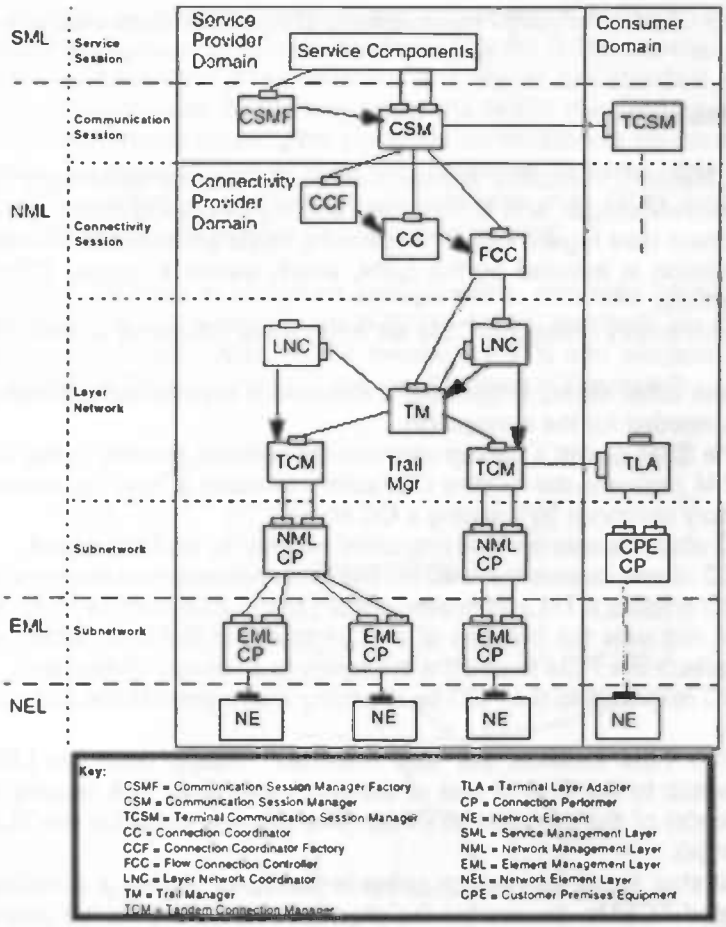


Figure 4-8: Connection Management Architecture (Computational Viewpoint)

The conceptual groupings and the associated objects are listed below, together with a description of the functionality of each object:

Communication session related objects

- *Communication Session Manager Factory (CSMF)* - creation of CSM objects and general communication session management (list sessions, acquire session control interfaces, delete sessions)
- *Communication Session Manager (CSM)* - set up and control and management of communication sessions
- *Terminal Communication Session Manager (TCSM)* - high level abstraction from an end-user terminal, responsible for establishing the nodal part of a stream flow connection

Connectivity session related objects

- *Connection Coordinator Factory (CCF)* - creation of CC objects
- *Connection Coordinator (CC)* - manipulation of the connectivity session associated with a connectivity session
- *Flow Connection Controller (FCC)* - control of the Network Flow Connection associated with a connectivity session

Layer network related objects

- *Layer Network Coordinator (LNC)* - establishment of connections in relation to a real network
- *Trail Manager (TM)* - manipulation of the associated trail
- *Tandem Connection Manager (TCM)* - manipulation of the associated tandem connections
- *Terminal Layer Adapter (TLA)* - set up, manipulation and deletion of network flow endpoints

Subnetwork related objects

- *Connection Performer (CP)* - management of subnetwork connections across one subnetwork

4.3.6 Binding Establishment

In the TINA Network Resource Architecture connection establishment is initiated by the Service Session Manager, and is controlled entirely by objects residing in the Connectivity Provider Domain (see Figure 4-8). The following steps are taken to set up a connection:

1. A connection is initiated by the SSM, which issues a 'create CSM' request to the CSMFactory.
2. The CSMFactory creates a CSM an returns the reference to that CSM object to the SSM.
3. When the CSM object is created, it chooses a connection provider and locates the TCSMs needed for the connection.
4. Next, the SSM issues a 'set-up connectivity session' request to the CSM.
5. The CSM requests the related CCFactory to setup a new connectivity session. The CCFactory responds by creating a CC object.
6. This CC object issues an 'add branches' request to an FCC object.
7. The FCC object locates the LNC for this layer network and requests a trail setup.
8. The LNC creates a TM and requests this TM to add branches to the new trail.
9. The TM requests the creation of a TCM object to the LNC, which creates the TCM and requests the TCM to add the branches to a tandem connection.
10. The LNC responds to the FCC by returning a reference to the trail control interface of the TM.
11. When the TCM receives the 'add branches' request from the LNC, it request an NFEP setup to the TLA of one of the endpoints. This TLA returns the name(s) and reference(s) of the selected NFEP(s). The same is done for the TLA(s) of the other endpoint(s).
12. The last step in the connection setup is the TLAs issuing a correlate request to the associated TCSMs, to connect the physical connections to the (higher level) stream flow endpoints.

As already mentioned, binding establishment is managed entirely by centralised objects. These objects use connection graphs (SSGs, LCGs and PCGs) which hold information about bindings and resources. This allows these centralised objects to make decisions about which configuration to use, and whether a requested QoS can be met. As is shown in **Error! Reference source not found.**, the TINA Network Resource Architecture provides objects for high-level stream binding management, as well as objects to manage low level connections between network elements. This 'total-control' approach allows a TINA NRA-system to manage every aspect of a binding.

4.4 The ITU-T H.323 Standard

4.4.1 Introduction

The ITU-T H.323 standard describes terminals, equipment and services for multimedia communication IP-based networks, without providing a guaranteed QoS. H.323 terminals and equipment may carry real-time voice, data and video, or any combination, including videotelephony.

The H.323 standard is widely adopted by commercial vendors, and is the standard technology for multimedia conferencing over LANs without guaranteed QoS. It is a 'parapet-standard', which 'borrows' the functionality provided by a number of other ITU-T standards, like the H.245 standard which describes signalling procedures between endpoints. Because the H.245 standard is also used by related standards (like the H.310 standard, which defines multimedia communication over ATM networks and the H.324 standard, which defines multimedia communication over PTSN networks), the H.323 standard can interwork with these other standards, thereby creating the possibility for multimedia communication across different networks. To establish connections with endpoints that use other standards (for example the H.320 standard for multimedia communication over ISDN, which uses H.242 for signalling), a *gateway* can be used to translate the signalling.

H.323 terminals may be used in multipoint configurations, and may interwork with H.310 terminals on B-ISDN, H.320 terminals on N-ISDN, H.321 terminals on B-ISDN, H.322 terminals on guaranteed QoS LANs, H.324 terminals PSTN and wireless networks, and V.70 terminals on PSTN. These configurations are displayed in Figure 4-9:

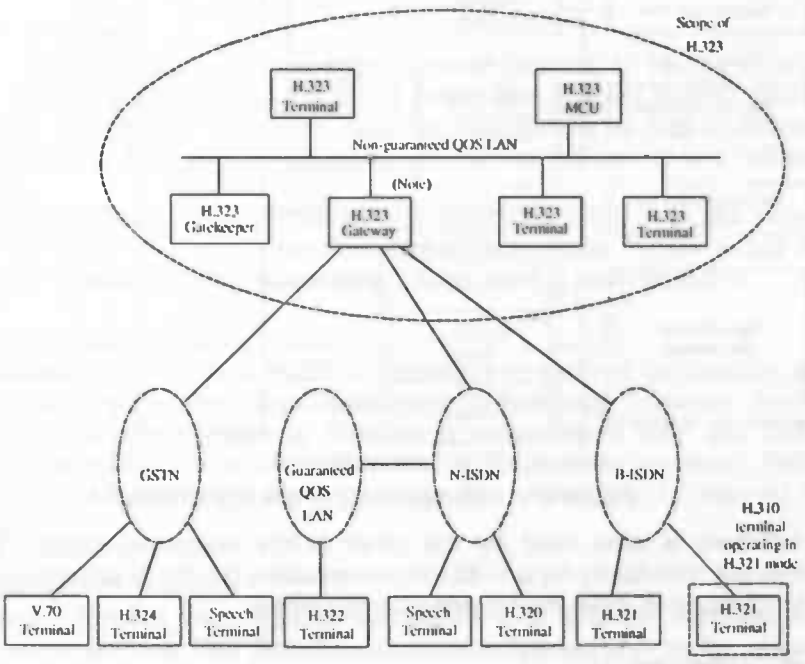


Figure 4-9: Interoperability of H.323 terminals

4.4.2 Architecture

The H.323 standard defines four entities from which functionality and protocols are described: Terminals, Gateways, Gatekeepers and Multipoint Control Units [De Muijnck, 1997]. An overview of the architecture of these entities is given in Figure 4-10.

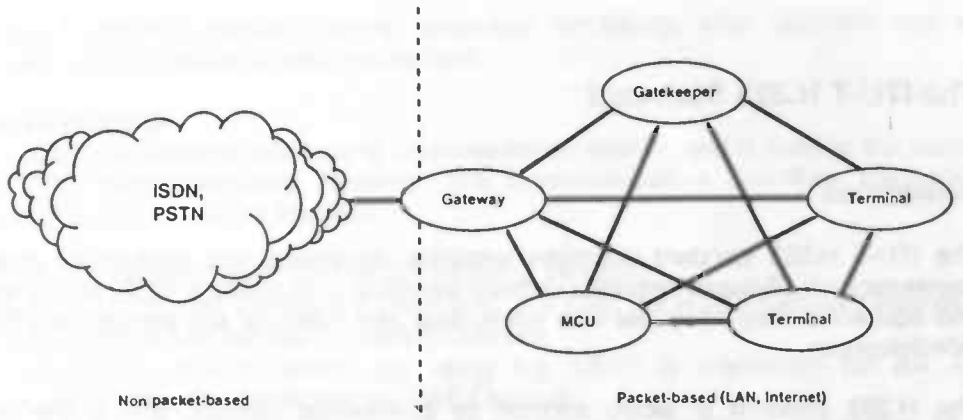


Figure 4-10: Different entities in the H.323 architecture

The entities Gateway, Gatekeeper, Terminal and MCU all have their own network connection to the LAN. On the left side, a Switched Circuit Network like ISDN or the PSTN (Public Switched Transport Network; the 'traditional' telephone network), is displayed. Through the Gateway, a connection can be established with terminals connected to the Switched Circuit Network. The different entities are explained in the next Sections.

4.4.3 The H.323 Terminal

The components of an H.323 terminal are shown in Figure 4-11. The diagram shows the user equipment interfaces, video codec, audio codec, telematic equipment, H.225.0 control layer, system control functions and the interface to the LAN.

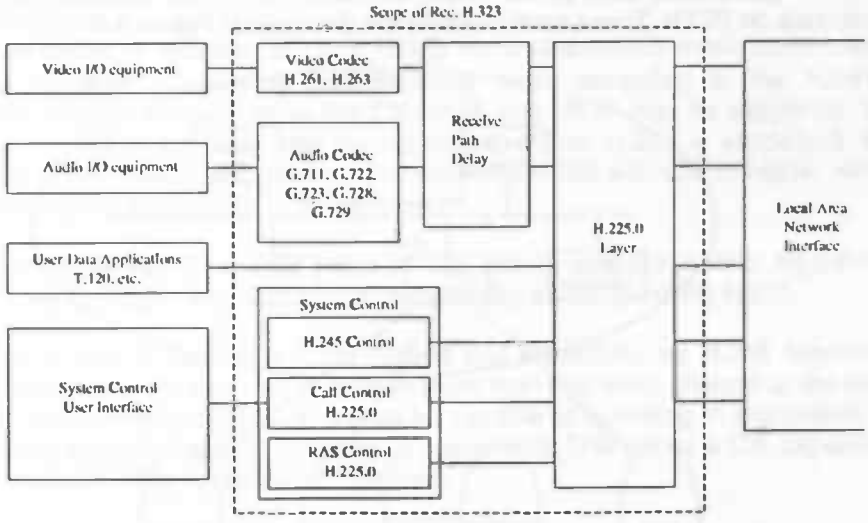


Figure 4-11: Components of a H.323 terminal

This architecture is also used by the other H.32*x* series standards. The following components are mandatory for a H.323 implementation (in ITU-T recommendation terms, a H.323 terminal 'shall' have the following components):

- System Control Unit
- H.225.0 Layer
- Network Interface
- Audio Codec

The Video Codec and User Data Applications components are optional. Figure 4-12 shows the components of a H.323 terminal in a 'protocol-stack', emphasising on the coherence between the components and protocols. The elements in the protocol stack are described in the paragraphs below.

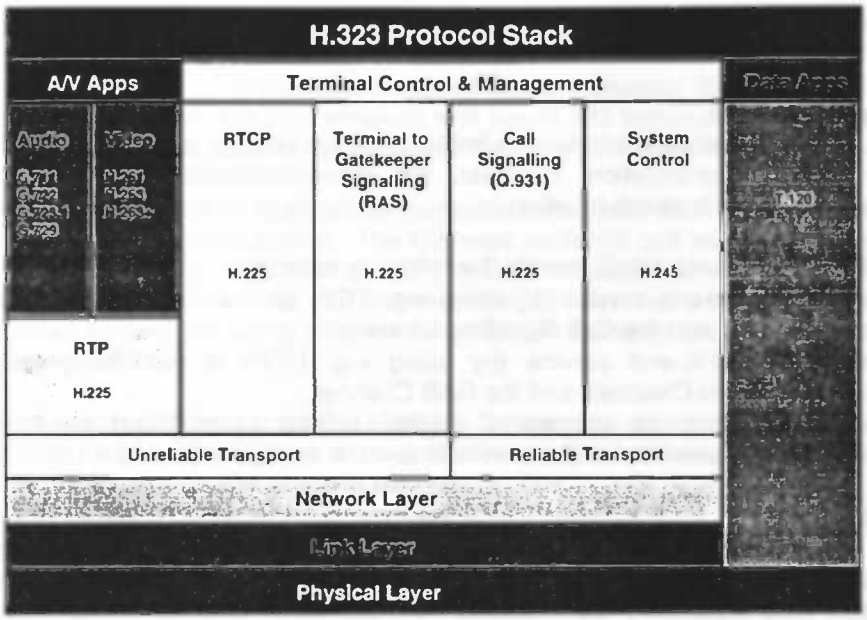


Figure 4-12: H.323 Protocol Stack

System Control Unit

The system control unit provides signalling for proper operation of the H.323 terminal. It provides the following facilities: call control, capability exchange (configuration negotiation), signalling of commands and indications and messages to open and fully describe the contents of logical channels (stream connections).

The system control unit consists of the following elements:

H.245 Control function

The H.245 control function uses the H.245 Control Channel to carry end-to-end control messages governing operation of the H.323 entity. The protocol entities specified by the H.245 include capabilities exchange, opening and closing of logical channels, mode preference requests, flow control messages and general commands and indications.

H.245 signalling is established between two H.323 terminal, a H.323 terminal and a Multipoint Controller or a H.323 terminal and a Gatekeeper. There exists exactly one H.245 Control Channel for each connection the endpoint is participating in.

H.225.0 RAS Control function

The RAS control function uses H.225.0 messages to perform registration, admissions, bandwidth changes, status and disengage procedures between terminals and Gatekeepers. This RAS Signalling Channel is independent from the Call Signalling Channel and the H.245 Control Channel. When no Gatekeeper is present, RAS Control is not used. The RAS Signalling Channel is the first channel to be opened in a H.323 connection.

H.225.0 Call Control function

The call control function uses H.225.0 call signalling to establish a connection between two H.323 terminals. This Call Signalling Channel is opened prior to the establishment of the H.245 Channel and any other logical channels between H.323 terminals, and is independent from the RAS Channel and the H.245 Control Channel.

The Call Signalling Channel can be opened between two H.323 terminals, or between a terminal and a Gatekeeper.

H.225.0 Layer

The H.225.0 layer formats the transmitted video, audio, data and control streams into messages for output to the network interface and retrieves the received video, audio, data and control streams from message which have been received by the network interface. In

addition, it performs logical framing, sequence numbering, error detection and error correction as appropriate to each media type

Network Interface
The Local Area Network Interface is implementation specific and is outside the scope of the H.323 recommendation. However, the recommendation describes the required functionality of the Network interface.

The Network interface 'shall' provide the following services:

- Reliable end-to-end service (by using e.g. TCP) for the H.245 Control Channel, the Data Channels and the Call Signalling Channel.
- Unreliable end-to-end service (by using e.g. UDP) is mandatory for the Audio Channels, Video Channels and the RAS Channel.

These services may be simplex or duplex, unicast or multicast depending on the application, the capabilities of the terminals and the configuration of the LAN.

Audio Codec
Each H.323 terminal is obliged to have an audio codec. The G.711 audio codec, which is capable of encoding and decoding A-law and μ-law audio (and speech) is mandatory. A terminal may optionally be capable of encoding and decoding speech using Recommendations G.722, G.723, G.728, G.729 and MPEG 1 audio. The audio codec used by the encoder shall be derived during the capability exchange using H.245.

An H.323 terminal should be capable of sending more than one audio channel at the same time, and should be capable of asymmetric operation, e.g. it should be able to send G.711 and receive G.728 if it is capable of both.

Video Codec
The video codec is optional. When a H.323 terminal provides video communications, it shall be capable of encoding and decoding video according to the H.261 QCIF[1]. Optionally, other video resolutions like CIF, 4CIF and 16CIF may be supported. The more advanced H.263 codec may also be supported. When H.263 is supported, the QCIF resolution is mandatory, and if a terminal supports H.263 with CIF or higher resolution, it shall also support H.261 CIF.

Configuration parameters like the video bit rate, picture format and other algorithm options accepted by the codec are defined during capability exchange using H.245.

A terminal is free to use other video codecs and resolutions via H.245 negotiation. The terminal may optionally transmit or receive more than one video channel at the same time, possibly using different codecs, and shall be capable of operating in asymmetric video bit rates, frame rates and picture solutions (if supported). This allows a CIF capable terminal to transmit QCIF while receiving CIF pictures.

User Data Applications
One or more data channels are optional. A data channel may be unidirectional or bi-directional depending on the requirements of the data application. Recommendation T.120 is used to provide data interoperability between H.323 terminals and between H.323 terminals and H.320 or H.310 terminals. The use of data channels is negotiated via H.245 negotiation.

A T.120 connection may be opened prior or after an H323 connection is established. When the H.323 connection is established first, normal call set-up procedures are followed. In the case where the T.120 connection is established first, normal call set-up procedures are followed, and the T.120 connection is associated with the H.323 connection after it is established.

---

[1] CIF, or Common Intermediate Format, is a standard format for video images. The size is 352*288 pixels. There are a number of derived formats, like QCIF which is a quarter the size of CIF, and 4CIF, which is four times the size of CIF.

4.4.4 Gateway

A Gateway provides communication to Switched Circuit Networks (SCNs), like ISDN and PSTN. It is used to connect H.323 terminals with non-H.323 terminals on these networks. Figure 4-9 shows the possible configurations for inter-network communication.

A Gateway is responsible for appropriate translations between transmission formats and between communication procedures. The Gateway performs call set-up and clearing on both the LAN side and the SCN side. It also performs translation of video, audio and data formats, so that a H.323 terminal can communicate with terminals that are connected to different networks and are using different protocols. A Gateway is defined by the ITU-T H.246 recommendation.

When a call is incoming from the SCN side, the Gateway co-operates with a Gatekeeper. One of the tasks of a Gatekeeper is to determine to which terminal the call has to be routed. The Gatekeeper is discussed in the next Section.

4.4.5 Gatekeeper

A Gatekeeper is not mandatory for a H.323 environment, but when a Gateway is used, the use of a Gatekeeper is strongly recommended. A Gatekeeper performs three important tasks:

- Address translation
- Bandwidth management
- Access control

More than one Gatekeeper can be present in a network. Each Gatekeeper controls a *zone*. All entities in this zone are managed by the Gatekeeper that controls the zone. A zone can contain terminals (Tx), Gateways (GW) and MCUs (MCU), but it must contain at least one terminal. A zone is independent of network topology, so a zone can exist of (parts of) different segments, connected through routers (R). Figure 4-13 shows an H.323 zone.

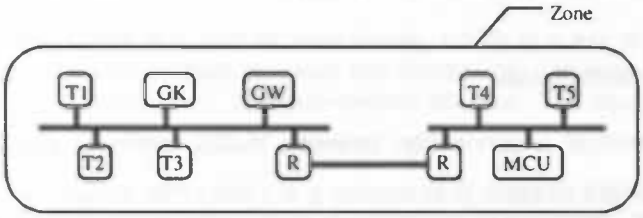


Figure 4-13: H.323 Zone

The Gatekeeper performs address translations from alias addresses (like network aliases or e-mail addresses) to network addresses (the IP-address). It also takes care of routing incoming calls from the Gateway to the correct H.323 terminal in the zone.

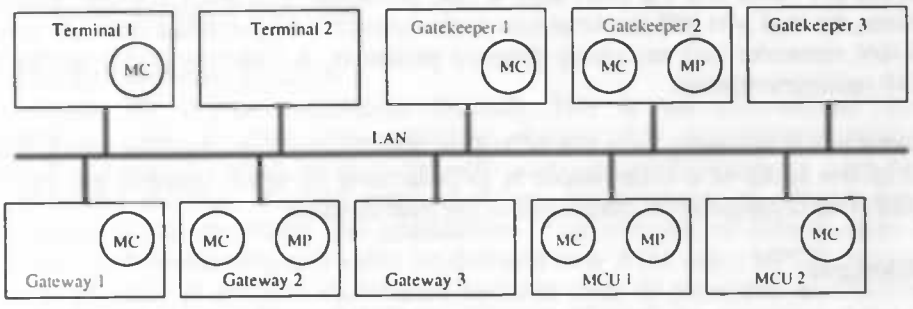
A second, optional function of the Gatekeeper is bandwidth management. A Gatekeeper can control bandwidth use per terminal, but also the total amount of bandwidth used for video-conferencing in the zone it controls. The H.225 RAS protocol is used to control the terminals in the network.

The third function of the Gatekeeper is access control. The Gatekeeper determines who is granted or refused access to the network. Criteria for granting or refusing access are not defined in the H.323 standard.

As stated before, a Gatekeeper is not mandatory in a H.323 environment. But when a Gatekeeper is present, all terminals in the zone controlled by the Gatekeeper have to obey the Gatekeeper, which makes a Gatekeeper an excellent tool for network management [De Muijnck, 1997]. It is also foreseen that the functionality of a Gatekeeper will be combined with for example a Gateway or MCU.

4.4.6 Multipoint Control Unit

A multipoint control unit (MCU) facilitates connections between three or more terminals. An MCU supports centralised and decentralised sessions. In a centralised session all communications are routed via the MCU, in a decentralised session communications are routed directly to each participant. Centralised and decentralised sessions can be mixed in a single session. The advantage of a decentralised session is the use of multicast techniques, which reduces bandwidth use.



NOTE – Gateway, Gatekeeper and MCU can be a single device.

Figure 4-14: Possible locations of MCs and MPs in a H.323 environment

An MCU consists of a Multipoint Controller (MC) and zero or more Multipoint Processors (MPs). The MC provides control functions in multipoint conferences. For example, it carries out the capabilities exchange with each endpoint in the multipoint conference. An MP processes the audio and video streams by mixing channels or by switching between channels. The selection whether to use switching or mixing is determined through H.245 control. An MP may also provide algorithm and format conversion, allowing terminals to participate in a conference at different networks. In an H.323 environment MPs are available supporting audio, video and T.120 data. Figure 4-14 shows the possible locations of MCs and MPs in a H.323 environment.

The interfaces (APIs) of the MC and MP components are accessible from an external program. Only an MCU is callable. The MCU communicates with terminals through H.245 messages.

4.4.7 Connection Establishment

The establishment of a connection between H.323 terminals can be divided in three stages:

1. Establishment of an initial connection
2. Initial communication and capability exchange (QoS and configuration negotiation)
3. Establishment of the actual stream binding (i.e. the audiovisual connection)

Each stage is described below:

1. Establishment of an initial connection

To establish an initial connection (used to exchange configuration and control information), the call control messages defined in the Q.931 call signalling protocol (which is part of the H.225.0 call control protocol) are used. The terminal sends a message to the 'well known' TSAP (Transport layer Service Access Point) identifier of the other terminals. For this identifier, port address 1720 is reserved.

When a Gatekeeper is present on the network, an Admission Request signal is first sent to this entity. The Gatekeeper returns an Admission Confirm signal when it allows the connection, or an Admission Reject signal when it rejects the connection.

When the terminal is allowed to communicate, a Set-up message is sent to the peer terminal. The peer terminal returns a Call Proceeding message, to indicate that the terminal is ready. The terminal signals the user that a connection is being requested. The

user can decide to accept or reject the call. During the signalling and acceptation or rejection of the call, alerting messages are sent to the calling terminal. When the connection is accepted a Connect message is sent to the calling terminal.

2. Initial communication and capability exchange
The connection established in the previous stage is a reliable connection, which uses the TCP protocol. In this stage this connection is used for initial communication and capability exchange. The terminals involved in the connection exchange the capabilities they support with respect to e.g. the codecs to be used, the resolution of the video image, etc., by using messages and properties defined in the H.245 system control protocol. This exchange results in a set of capabilities supported by all terminals. This information is used to set-up the stream bindings.

An important property is the assignment of dynamic portnumbers to each channel. These portnumbers can differ per connection which may cause problems when trying to establish a H.323 connection through a firewall. Because of this dynamic portnumber assignment it is difficult for firewalls to monitor the H.323 packets.

3. Establishment of the actual stream binding
Now the configuration is determined, the actual stream binding is established, by starting the desired audio and video streams. These streams are encoded, divided in packets by the H.225.0 layer and transmitted. The receiver restores the order of the received packets, restores the streams and decodes the contents. As can be seen in Figure 4-12, the H.323 standard uses the RTP protocol for the transmission of the multimedia data. This protocol is discussed in Section 3.4.

4.5 Other Solutions and Standards

In this section a number of other solutions and standards are briefly discussed. These solutions and standards are selected on basis of the criteria mentioned in Section 4.1, and are expected to form a representative view of 'what is available'.

4.5.1 Java Media Framework

The Java Media Framework is a part of Java Media, which is a set of APIs (Application Programming Interfaces) developed to meet the increasing demand for multimedia by providing a unified, non-proprietary, platform-neutral solution. The standard is developed by Sun and Intel.

The Java Media Framework API (JMF) is a collection of classes that enable the display and capture of multimedia data within Java applications and applets. JMF addresses the following multimedia requirements:

- *Media display* - playback of local and network multimedia data within an application or applet
- *Media capture/creation* - the ability to record, save and transfer data through local capture devices, such as microphones and cameras
- *Conferencing* - full-duplex, streaming media with real-time constraints

The Java Media Framework APIs are released in stages. The first API release includes the Java Media Player.

Java Media Player
The Java Media Player provides a set of high-level interfaces for the reception and playback of arbitrary time-based media such as MPEG-1, MPEG-2, Quicktime, a number of other popular audio and video formats, and real-time streaming audio/video. Media data can originate from a variety of sources, like files, URLs, or streaming sources like video-on-demand servers or RTP connections.

Remarks

Although not completely released (the conferencing API which is expected to be the most interesting, is not released yet) the Java Media Framework looks very promising. The Java language has become very popular and implementation of ODEs in Java is already possible (an example of such an implementation is IONA's OrbixWeb). When the JMF is finished it should be investigated how it can be integrated with an ODE. It is expected that the combination of Java, multimedia and an ODE has a great potential, because it makes it possible to create distributed, platform independent multimedia applications. The JMF is also very easy to extend, making it possible to integrate for example QoS capabilities.

4.5.2 DAVIC DSM-CC, MPEG-4 and DMIF

The Digital Audio-Visual Council

DAVIC stands for Digital Audiovisual Council. This is an international non-profit organisation founded in 1994 and consisting of over 200 corporations representing business that have a strong relationship with audio-visual applications and services. DAVIC promotes the idea of using open interfaces and protocols to achieve maximum interoperability across countries and applications. The activities of DAVIC were mainly concentrated on broadcasting and video on demand over ATM using MPEG, but recently, attention is focused to support multimedia streams over IP networks and facilities for multimedia communication [DAVIC CFP12], [DAVIC CFP13]. An overview of the DAVIC system can be found in [Leydekkers, 1997].

This thesis focuses primarily on the control and management of multimedia streams. Therefore, the Session & Transport Service Layer (addressed by interfaces called S2 and S3) is of primary interest here. This layer is implemented by the Digital Storage Media Command & Control (DSM-CC) protocol which is part of the MPEG specification. In the DAVIC 1.0 specification the version incorporated in MPEG-2 is used. In this Section MPEG-4 and the Delivery Multimedia Integration Framework (DMIF) which is integrated in the MPEG-4 specification are also treated, because of the close relationship with DSM-CC and interesting features it adds to MPEG-4 with respect to stream control and management.

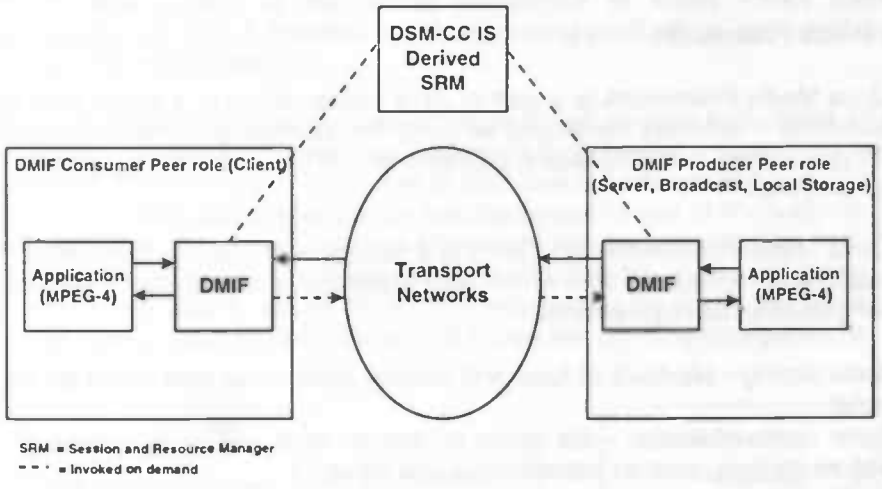


Figure 4-15: Relations between DSM-CC, DMIF and MPEG-4

DSM-CC

The DSM-CC component is mainly used to set-up and control multimedia streams for broadcasting and video-on-demand over ATM using MPEG-1 and MPEG-2. DSM-CC consists of two separate sets of protocols: the User-to-User protocols (DSM-CC UU), used for the establishment of multimedia streams, and the User-to-Network protocols (DSM-CC UN), used for communication between user and server.

The DSM-CC UU-protocols define DSM-CC user-to-user signalling and are used for user-to-user application control. DSM-CC UU provides standardised access to multimedia objects or applications, without prior knowledge of their location.

The DSM-CC UN-protocols are used to control multimedia streams. The following functions are included: [Noorlander, 1998]:

- *Configuration management* - used to provide the users with DSM-CC and network specific parameters which are needed for communication with the SRM (Session and Resource Manager) and for the initiation of the first UU session
- *Session management* - used for resource negotiation and to minimise the effect of network failures
- *Switched Digital Broadcast - Channel Change Protocol* - defines how the signalling must be performed between an InterWorking Unit (IWU) and the end user in such an architecture. This falls out of the scope of this thesis.

In the OMG Control and Management of Audio/Video Streams specification relations between this specification and DSM-CC are described. It is stated that the are areas of overlap with the DSM-CC User-to-User part. A DAVIC stream could be managed as a flow within the architecture proposed by the OMG A/V spec.

MPEG-4

MPEG-4 is a new standard developed by the Motion Picture Experts Group, a technical committee which develops standards for coded representation of moving pictures, associated audio, and their combination when used for storage and retrieval on digital storage media [Lu, 1996].

MPEG-4 builds on and combines elements from three fields: digital television, interactive graphics and the World Wide Web. It is different from the previous MPEG standards in that it concentrates on *coding of audiovisual objects* instead of coding to achieve low bitrates. The standard is meant to provide a generic toolbox for many different kinds of applications (e.g. conversational, interactive and broadcast).

The goals of MPEG-4 can be summarised by the following keywords:

- *Interactivity* - MPEG-4 information is content based and can be randomly accessed
- *Integration of natural and synthetic material* - synthetic (e.g. 3D objects) and natural objects (e.g. speech and video) can be mixed together in the same scene
- *Information can be accessed anywhere* - information can be accessed from mobile networks, from error-prone environments, from heterogeneous networks, while providing scalability and QoS specifications
- *Incorporation of Intellectual property rights* - incorporate facilities for identification and protection of property rights for content

These goals are achieved by providing standardised ways for:

1. The coded representation of units of aural, visual or audiovisual content, called "audio-visual objects" or AVOs;
2. The way individual AVOs are composed in a scene;
3. The way AVOs are multiplexed and synchronised, so that they can be transported over network channels providing a QoS appropriate for the nature of the specific AVOs or user requirements;
4. A generic interface between the application and the transport mechanisms;
5. The way the user interacts with the scene (e.g. changing the viewpoint) and the individual objects in a scene (e.g. clicking on an object to inquire about its features);
6. The projection of the AV scene so composed on the desired viewing/hearing point

MPEG-4 uses the DMIF standard for the set-up of connection channels and to provide network transparency to applications which use MPEG-4. For more information on MPEG-4, see the MPEG-4 Web-page (URL: http://drogo.cselt.stet.it/mpeg).

DMIF

DMIF (Delivery Multimedia Interaction Framework) is a component of the MPEG-4 standard. DMIF addresses operations of multimedia applications over interactive networks, in broadcast environments and from disks. The DMIF architecture is such that applications which rely on DMIF for communications do not have to be concerned with the

underlying communications method. The implementation of DMIF takes care of the network details presenting the application with a simple interface.

An MPEG-4 application can request from DMIF the establishment of channels with a specific QoS and bandwidths for each elementary stream. DMIF ensures the establishment of the channels with the specified bandwidths while preserving the QoS over a variety of intervening networks between the endpoints involved in the connection.

Remarks
The DSM-CC protocols are used by the DAVIC system to set-up and control multimedia streams. DAVIC used to focus primarily on broadcasting and video-on-demand over ATM using MPEG. Recently, moves toward supporting IP networks and integrating multimedia communications are made. These moves can provide interesting results, especially in interworking between standards for set-up and control of multimedia streams over heterogeneous (ATM and IP) networks. This expectation is strengthened by the fact that the OMG Control and Management of A/V Streams specification mentions relationships with DSM-CC in overlapping functionality.

DMIF features set-up and control of multimedia streams and providing network transparency to applications that use MPEG-4 could be of interest for the integration of multimedia in ODEs. However, DMIF is a new development and little information on this standard is yet available.

MPEG-4 is a new standard which combines elements of the fields of digital television, interactive graphics and the World Wide Web by focusing on coding of audiovisual objects, instead of coding to achieve low bitrates. The MPEG-4 standard is expected to be a powerful medium for the delivery of multimedia content, which could be used in an ODE to deliver interactive content (e.g. for marketing purposes).

4.5.3 Meetingpoint

White Pine's Meetingpoint is the server component of White Pine's client/server videoconferencing solution. It provides a 'virtual meeting point' where groups of users can communicate and collaborate.

At this moment, Meetingpoint is the only software-based solution to facilitate Wide Area Network (WAN) point-to-multipoint and multipoint-to-multipoint connections using the H.323 standard. The software is optimised to minimise the amount of bandwidth used by using advanced routing techniques and by making extensive use of IP-multicast. It is also possible to establish connections between different kinds of equipment (like Intel Proshare or PictureTel) and software (like Netmeeting or CU-SeeMe) connected using different kinds of connections (like a LAN or ISDN) The requirement for this is that the equipment and software must support H.323. Figure 4-16 shows two possible configurations.

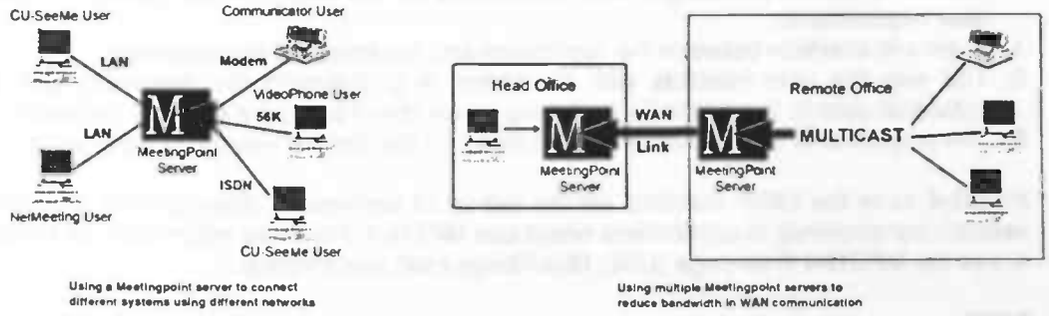


Figure 4-16: Examples of Meetingpoint configurations

Multiple Meetingpoint servers can co-operate by distributing the communications load, and by reducing the network load. This is achieved by putting a Meetingpoint server at each network endpoint and by routing the inter-network communications via these

servers. The servers use IP-multicast where possible, thereby achieving a significant reduce of workload.

Remarks
Meetingpoint is a software-based multipoint server which is able to connect various types of equipment across a WAN. Advanced bandwidth reduction techniques are used to reduce network load. Another interesting feature is that multiple Meetingpoint servers can be used to reduce each server's workload. The concepts used by Meetingpoint are very interesting to use in an ODE. Especially the bandwidth reduction techniques are very interesting, and it is expected that these techniques can be easily implemented by using ODE facilities (an ODE supports techniques like replication, so uniform techniques can be used to implement this facility). For more information about Meetingpoint, see White Pine's Web-site (URL: http://www.wpine.com/Products/Meetingpoint)

4.5.4 Netmeeting

Netmeeting [Netmeeting] is Microsoft's videoconferencing and dataconferencing tool. It supports point-to-point videoconferencing using the H.323 standard and multipoint dataconferencing using the T.120 standard. It also supports LDAP (Light Directory Access Protocols) [Rose, 1997] directory services to create directories of current Netmeeting users. Figure 4-17 shows an overview of the Netmeeting architecture.

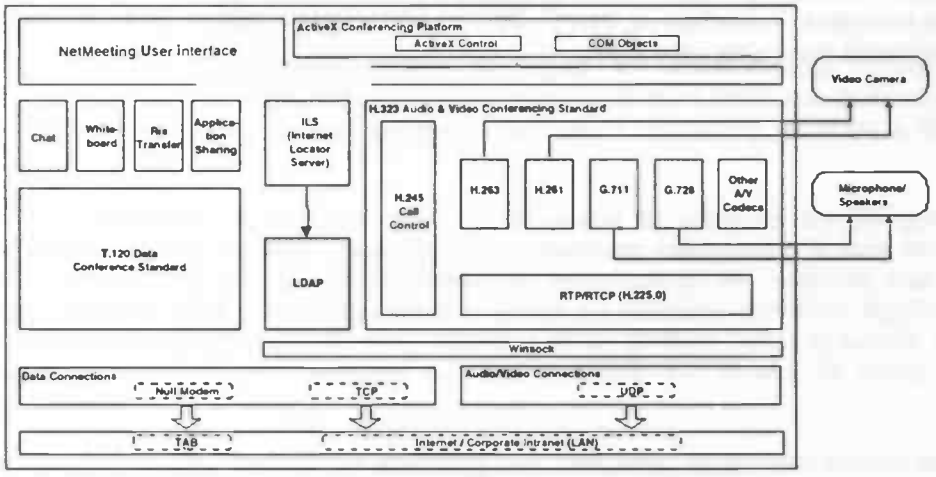


Figure 4-17: Overview of the Netmeeting architecture

Netmeeting facilities can be easily integrated in other applications because it is built of ActiveX components. This technology makes it possible to use components of the Netmeeting platform in other applications.

Remarks
Netmeeting is a software-based H.323 implementation, that serves as a general, windows-based platform for H.323 communication. Its added features like LDAP and the use of ActiveX components make it an interesting platform. Because it is heavily supported by Microsoft, it is expected to speed up the acceptation of the H.323 protocol as a standard protocol for video-conferencing. The platform is already supported by many hardware vendors selling audiovisual hardware. A Software Developers Toolkit (SDK) is also available to support software development. Despite these advantages, Netmeeting's largest disadvantage is that it is heavily based on the Windows operating system, so it is not portable. This is an important disadvantage when integrating Netmeeting into an ODE; it can then only be used on a Windows-based system.

When Netmeeting is integrated in an ODE, its primary use will be as a H.323 terminal, communicating with ODE servers which provide multimedia stream control and management features. It is interesting to investigate how to use Netmeeting as H.323 endpoint-component in the OMG Control and Management of A/V Streams specification.

For more information about Netmeeting, see Microsoft's Web-site (URL: http://www.microsoft.com/netmeeting).

4.6 Summary

In this Chapter a number of solutions and standards for distributed multimedia systems are discussed. The goal of the Chapter is to give a good overview of which solutions and standards are available. Therefore, a survey of existing solutions and standards is made. Solutions and standards are selected based on a number of criteria (of which the most important is that the solution or standard should be widely used and that the solution or standard should be integrated or easily be integrated with and open distributed environment).

Three standards are discussed extensively, because they best meet the criteria posed and are closest to the binding object concept: the OMG Control and Management of Audio/Video streams specification, the TINA Network Resource Architecture and the ITU-T H.323 standard. These standards were selected because they differ in both design paradigm (object-centered vs. protocol-centered) and control method (central responsibility vs. distributed responsibility).

The other solutions and standards selected (DAVIC DSM-CC, DMIF and MPEG-4, Java Media Framework, White Pine Meetingpoint and Microsoft Netmeeting) were discussed less extensive. Nevertheless, they provide a good insight in what work is carried out in the multimedia communication area.

5 A Reference Model for the Comparison of Multimedia Stream Binding Solutions

5.1 Introduction

From the previous chapters it can be concluded that multimedia services are a valuable addition to ODEs. This has been recognised by standardisation bodies and commercial vendors of multimedia applications and ODEs, resulting in a number of specifications and implementations for multimedia integration in ODEs. Examples of these are the OMG Control and Management of Audio/Video Streams of the Object Management Group, and the TINA Network Resource Architecture (see Chapter 4). Also, other standards which are not specifically designed to be used in ODEs can be added. An example of such a solution is the ITU-T H.323 standard (see Chapter 4).

The specifications and implementations developed by standardisation bodies and commercial vendors have resulted in a large number of different solutions. In most cases these solutions have very distinct designs. This makes comparing solutions a difficult task.

However, comparison of different solutions is useful to examine advantages and disadvantages of the different solutions, and to examine interworking issues between these solutions. This can be accomplished by investigating the relations that exist between solutions, and by investigating how (parts of) solutions can work together, or perhaps be replaced by (parts of) other solutions which perform better at certain tasks. Therefore, a method was searched for that provides the means to make such comparisons.

In the literature, such a method for analysing and comparing multimedia stream binding solutions was not found (See Section 5.1.2 on 'Survey of existing methods' below). To make the evaluation and comparison of different multimedia stream binding solutions possible, it was decided to develop a reference model which can be used to analyse a solution from the ODP-RM Information and Computational Viewpoints, by modelling each solution using UML diagrams, and by addressing reference points in these diagrams. The resulting Information analyses and Computational specifications can then be used to identify the strong and weak points in each solution, and to make a comparison of (parts of) the solutions. This process is shown in Figure 5-1.

5.1.1 Goal of the Reference Model

An issue left open in the literature is the provision of a solid, practical reference model which can be used to model a multimedia stream binding solution, to make a strength/weakness analysis of this model, and to be able to make meaningful comparisons of this analysis with the strength/weakness analyses of other solutions. The goal of the reference model can be summarised as:

> *'To provide a solid, practical reference model to make strength/weakness analyses of multimedia stream binding solutions, which can be used for the comparison of multimedia stream binding solutions.'*

The added value of such a reference model exists in its underlying basis. Instead of just presenting a checklist with a number of requirements, and to leave the analysing-process

open (the process commonly followed by other authors, see Section 5.1.2), it is clearly defined which abstraction levels should be used, how a multimedia stream binding solution should be modelled before it is analysed, and to which items of an analysis a reference point applies. This makes it possible to make comparisons between possibly very distinct solutions, or parts of those solutions, while avoiding the risk of making illegal comparisons.

5.1.2 Survey of Existing Methods

A structured approach for the comparison of multimedia stream binding solutions was not found in the literature. In some cases, a checklist with requirements which a multimedia stream binding solution should meet was found [Leydekkers, 1997], [Blair, 1998], [Muhlhauser, 1996]. These checklists mostly address aspects like Quality of Service management, multiparty connections, etc.

Most authors agree on the need to add multimedia support to distributed processing environments. Leydekkers [Leydekkers, 1997] and Blair [Blair, 1998], for example, do suggestions for additions to the ODP Reference Model to add the support of multimedia. The general approach followed by these authors is to add support for multimedia to RM-ODP, and to use this extended reference model as a starting point to incorporate multimedia support into existing distributed processing environments, or to the develop a new programming model for multimedia (in the case of [Blair, 1998]).

Other authors focus primarily on system requirements, like the hardware and network requirements of the systems involved in a multimedia connection (see e.g. [Lu, 1996]). These issues are of course of great importance (multimedia applications pose certain minimum requirements on the hardware used), but they are not of key importance here.

5.2 The Reference Model

This section discusses a number of important aspects of the reference model. The stages of the reference model are discussed in the next sections.

5.2.1 Applying the ODP-RM Information and Computational Viewpoints

The ODP-RM Information and Computational Viewpoints are used to analyse each solution. This is carried out by making a model of each solution for these viewpoints. These models are constructed by using the object oriented modelling language UML (Unified Modeling Language). See Appendix A for an overview of UML.

The Information and Computational Viewpoints are chosen because the Information Viewpoint can be used to make an information model of a solution, which provides useful information about the information objects and the information flows between the other components of the solutions. The model created from the Computational Viewpoint provides useful information about the different components of a solution, and about the relation between those components.

5.2.2 Structure of the Reference Model

The structure of the reference model is shown in Figure 5-1. The reference model describes a process consisting of four stages. In the first stage, an Information Model and a Computational Specification are constructed. In the next stage, reference point are addressed in each diagram. These reference points point out important elements in each diagram.

In the third stage, analyses of the Information Model and the Computational Specification are made, by answering the issues addressed by the ODP-RM Information and Computational Viewpoints, respectively. These issues are answered by using the information provided by the reference points addressed in the previous stage. The last stage consists of converging these two analyses to one strength/weakness analysis. This

strength/weakness analysis is made by validating the analyses of the two ODP-RM Viewpoints with respect to the checklist discussed in Section 3.6.2. The checklist addresses the requirements of a multimedia stream binding solution.

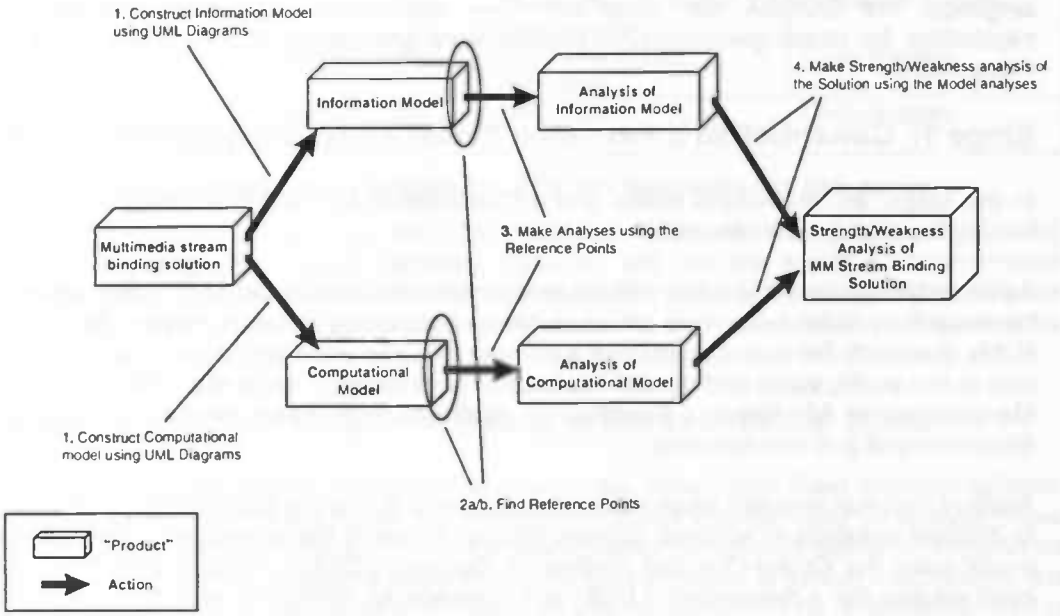


Figure 5-1: Structure of the Reference Model

5.2.3 The Level of Detail

The reference model uses the ODP-RM Information and Computational Viewpoints to make the analyses. But the level of detail in which a system is viewed from these Viewpoints is left open by the ODP-RM. A system can for example be looked upon as one component, or, when looking at a lower detail level, as a large number of related and interworking components.

It is very important to set clear which detail level is used when making the analyses, because when keeping the detail level too high, there might not be enough information available to make a good comparison, running the risk of making illegal comparisons due to this information shortage. On the other hand, when using a detail level that is too low, a lot of unnecessary work is done by creating unnecessary detailed and complex models.

The level of detail used for this reference model is by decomposing each solution into a number of separate components which each perform a separate, well defined task. Examples of such tasks are stream control and management, abstraction of hardware devices, QoS negotiation, and so on. For each component, the operations and properties of that component are also identified. This level of detail corresponds very good to the classes used in the Object Oriented paradigm, so an object oriented approach is a good method to construct the Information and Computational models.

This level of detail has proven to be sufficient to make good and legal analyses and comparisons, while preventing doing too much work by making unnecessary complicated models.

5.2.4 Choosing a OOA&D Tool to Construct the Models

There are already a number of Object-Oriented Analysis & Design tools available which support UML (such as Cayenne ObjectTeam, Rational Rose, and Paradigm Plus), and it is expected that most tools will support UML in a short time [Fowler, 1997].

For the construction of the diagrams the OOA&D tool *ObjectTeam* from Cayenne Software is used. This is a client/server based tool that uses a central repository, from which all clients can retrieve information. Strong points of ObjectTeam are its capabilities for version management, capabilities for generating source code for a large number of language, like CORBA IDL, Java and C++, capabilities for model checking, and capabilities for report generation for popular word processing packages like Word and Framemaker.

5.3 Stage 1: Construct an Information Model and a Computational Model

In this stage, an Information model and a Computational model of a multimedia stream binding solution are constructed.

As already observed it is often difficult to compare multimedia stream binding solutions, because they often have very different designs, covering different system levels. The H.32x standards for example address a number of audio and video codecs, and specify a way to mix audio, video and control information on a channel, while the OMG Control and Management of A/V Streams Specification abstracts from these issues and focuses on stream control and management.

Another common problem when comparing solutions is that the same terminology is used in different solutions to address different issues. To avoid these problems, the reference model uses the Object-Oriented Analysis & Design (OOA&D) method UML. To model each solution (for a description of UML, see Appendix A). Using this language, it is easy to model solutions based on the object-centered paradigm. Solutions based on the protocol-centered paradigm can also be modeled, because each system can also be modeled as a system built of (interacting) objects.

UML uses different types of diagrams to model a specification. Because there exists a relationship between the ODP-RM viewpoints and UML concepts, UML can be used to make the Information model and the Computational specification [MCI1, 1997]. Table 7 shows the relationship between the ODP-RM Information and Computational Viewpoints and UML Concepts (only the relations between these viewpoints are shown here, because they are used in this reference model). The relationship between ODP-RM viewpoints and UML concepts is described in more detail n Appendix B.

ODP-RM Viewpoints	ODP-RM concepts	UML concepts	UML diagrams
Information Viewpoint	invariant schema	package of references to class objects	Class diagram
	static schema	'snapshot' of the class diagram at a particular point in time, and, for objects with state machines, the current state of the state machine.	Class diagram and (possible) State diagram
	dynamic schema	UML state machine or a UML package of the state machines of several UML objects	State diagram
	information object	UML Association Class	Class diagram

continued on next page...

Computational Viewpoint	signal interface	UML message	Collaboration diagram
	operational interface	UML message (announcement) pair of UML messages (interrogation)	Activity diagram Sequence diagram
	stream interface	set of sets of UML messages	Class diagram
	computational object	UML Class	Class diagram

Table 7: Relationships between two ODP-RM Viewpoints and UML concepts

Note that both the Information and Computational Viewpoints can be described using UML class diagrams. These diagrams however, are not the same: The Information Viewpoint's class diagram focuses on the *information objects* of the system, while the Computational Viewpoint's class diagram focuses on the *computational objects* of the system.

5.4 Stage 2a: Address Reference Points in the Information Model

The ODP-RM Information Viewpoint focuses on the information flows between objects, and the contents of those flows. An information model can be used to analyse these information flows. An analysis of this information model can provide useful information on how, for example, the Quality of Service negotiation procedure is carried out between the endpoints, and what information is used in this procedure. The Information model addresses the following issues:

1. What are the information objects of the system?
2. What manipulations/processing can be performed on the information objects of the system?
3. What is the relationship between information objects?
4. What are the attributes of the information objects?
5. What are the rules and constraints for information manipulations?
6. What are the sources, sinks, and information flows in the system?
7. What semantics would be associated by a human with the information that is stored and exchanged between information objects?

Table 8: Issues addressed by the Information Viewpoint (adapted from [Leydekkers, 1997])

In order to answer these questions, the following reference points are addressed in the diagrams (Class diagram and State Machine diagram) of which the Information model is built. These reference points are selected to address interesting aspects for the analysis of multimedia stream control solutions (e.g. what QoS information is needed by what objects). The following two subsections show which reference points are addressed in each diagram. The number at the end of each reference point addresses the issue which can be answered by using the information provided by this reference point.

Reference points in the Class diagram
The following reference points can be addressed in the Class diagram:

1.1 Point out the information objects which hold Quality of Service information and/or configuration information (1)

1.2 Point out other information objects that might be of interest (1)

1.3 Point out information objects that contain other information objects (compound information objects) (3)

1.4 Point out which information objects are transmitted between what (computational) objects (6)

1.5 Point out the objects that use the QoS and/or configuration Information objects as source/sink (6)

1.6 Point out the objects that use other Information objects (addressed by Ref-pt. 1.5) as source/sink (6)

Reference points for the State Machine diagram
The following reference points can be addressed in the State Machine diagram:

2.1 Point out when QoS information objects are transmitted, and between which objects they are transmitted (6)

2.2 Point out when configuration information objects are transmitted, and between which objects they are transmitted (6)

2.3 Point out when 'other' information objects (addressed by Ref-pt 1.2) are transmitted, and between which objects they are transmitted (6)

2.4 Point out possible relationships between information objects, by locating choices in the state machine diagrams that are influenced by information objects (2, 3, 5)

2.5 Address the connection set-up phase (6)

2.6 Address the QoS and configuration negotiation and renegotiation phases (6)

2.7 Address the addition/removal of parties in a multiparty connection (6)

2.8 Address the phases to start/end a connection (6)

Additional Reference Points
The reference points addressed in this paragraph cannot be located in the Class or State Machine diagrams, because the information sources they point to is not part of those diagrams. However, they do make part of the Information Model.

3.1 List attributes of the QoS and configuration Information objects (4)
These attributes are generally found in tables in the specification.

3.2 List attributes of the 'other' Information objects (4)
List the attributes of the Information objects addressed by Ref-pt. 1.2

3.3 List the rules and constraints on the QoS and configuration Information objects (5)
These rules can generally be adapted from the tables described in Ref-pt. 3.1. These rules and constraints are closely related to the attributes, by addressing the bounds of these properties

3.4 Point out methods that perform manipulations and/or processing on the information objects (2)

5.5 Stage 2b: Address Reference Points in the Computational Model

The ODP-RM Computational Viewpoint focuses on structure of a system. Such a system is described as a set of interacting objects. A Computational model can be used to analyse this structure, in answering questions like 'what are the computational objects of the system, and how are they structured?' and 'what is the responsibility of these objects?'. This provides an insight in the design of the solution: what objects are used to encapsulate what functionality, and what are the interfaces of those objects. The analysis of the Computational model addresses the following issues:

1. What are the computational objects of the system and how are they structured?
2. What are the interfaces of the computational objects?
3. What operations can be invoked on the computational interfaces?
4. What is the role of each computational interface?
5. What behaviour is observable at the computational interfaces?
6. What environment constraints are associated with the computational objects and associated interfaces?
7. What interaction is possible between computational objects (interfaces)?

Table 9: Issues addressed by the Computational Viewpoint (adapted from [Leydekkers, 1997])

The Computational model consists of a Class diagram, a Collaboration diagram and a Sequence Diagram. The following subsections show which reference points are addressed in each diagram. The number at the end of each reference point addresses the issue which can be answered using the information provided by this reference point.

Reference points in the Class diagram
The following reference points can be addressed in the Class diagram:

4.1 Address all computational objects (with special attention to objects which carry out the following tasks: multiparty bindings, control, negotiation, abstraction of endpoints, abstraction of hardware devices) (1)
In most cases all classes in the class diagram are computational objects. The task of each class can mostly be derived from the name of the class, and by using extra information like the properties and methods and the specifications of these components

4.2 Address all public methods of each class which is addressed as a computational object by Ref-pt 4.1 (2, 3, 4, 7)
This reference point is used to point out the interfaces of each object (the operations which are accessible from the 'outside'), the role of each interface, and the constraints associated with each interface

4.3 Address exceptions of each operation, which identifies constraints on that operation (6)
This reference point is used to identify constraints of operations

Reference points in the Collaboration diagram
The following reference points can be addressed in the Collaboration diagram:

5.1 Address relations between the following objects: (1)
- control and negotiation objects
- control and multiparty binding objects
- control and abstraction of endpoint/hardware device objects

5.2 Address the operation sequences for the following operations, in point-to-point, point-to-multipoint and multipoint-to-multipoint connections: (5)
- connection set-up
- configuration negotiation
- QoS negotiation
- establishing the actual connection
- disconnecting

These operation sequences can be found in the collaboration diagram by looking where methods are called that perform these operations (these methods can be found by looking what methods are associated to classes that perform these operations in the class diagram).

Reference points in the Sequence diagram
The following reference points can be addressed in the Sequence diagram:

7.1 Address the relations between the following computational objects: (1)
- control and negotiation objects
- control and multiparty binding objects
- control and abstraction of endpoint/hardware device objects

These relations can be found by looking at the operations which are called between the objects, and the arguments which are used when calling these methods

7.2 Address the operations which are invoked to and from the computational objects (3)
These operations can be easily derived from the sequence diagram.

5.6 Stage 3: Analyse the Information Model and the Computational Model

Now that reference points have been addressed for all diagrams in both the Information model and the Computational model, the next step is to analyse the model. The purpose of these analyses is to get a good insight in the structure of a solution, the properties of a solution, and how certain features of a solution are designed (for example, how is the quality of service negotiation procedure designed, and what properties and constraints can be identified on this procedure?).

The analyses consist of the following elements:

- The UML diagrams created for the Information model and the Computational specification,
- The reference points assigned in the diagrams,
- A description of how these diagrams were constructed from the solution, and how the reference points were assigned to the diagrams,
- A description of how the issues addressed by the Information and Computational Viewpoints (see Table 8 and Table 9) can be 'filled in' by using the reference points assigned in the diagrams

In the next (and last) stage of the reference model, these analyses are used to make a strength/weakness analysis of the solution.

5.7 Stage 4: Converging Results - Making a Strength/Weakness Analysis

The last stage of the reference model consists of the making of a strength/weakness analysis of a multimedia stream binding solution, based on the analyses of the Information model and the Computational model created in the previous stage.

The target of the strength/weakness analysis is to point out the strong and weak elements in a multimedia stream binding solution, taking the checklist from Section 3.6.2 as a starting-point (See Table 10 below for an overview of this checklist). This checklist covers the requirements which a multimedia stream binding solution should meet. For each item in this checklist it is examined whether it is met by the multimedia stream binding solution. The reference points addressed in Stage 2 are used to provide the information needed to answer the items from the checklist.

The strength/weakness analysis of a multimedia stream binding solution consists of the following parts:

- A short description of the solution
- Analyses of the solution made from the ODP Information and Computational Viewpoints
- A description of how the solution meets the items mentioned in the checklist

1.	**Support for different protocols and codecs**
1.1	Does the multimedia stream binding solution support all important protocols and standards?
1.2	Does the multimedia stream binding solution accept any incoming multimedia flow, independent of its type?
1.3	Is the multimedia stream binding solution able to bind to any other interface?
2.	**Support for Quality of Service management**
2.1	Does the multimedia stream binding solution provide methods to avoid delay jitter?
2.2	Does the multimedia stream binding solution support bandwidth reservation and/or other resource reservation?
2.3	Does the multimedia stream binding solution provide negotiation of Quality of Service?
2.4	Does the multimedia stream binding solution provide the change of the Quality of Service provided?
2.5	Does the multimedia stream binding solution provide dynamic Quality of Service management, such as monitoring, maintenance, policing and renegotiation of the quality of service requested?
3.	**Support for multiple flow connections**
3.1	Is the multimedia stream binding solution able to bind to multiple flows?
3.2	Is the multimedia stream binding solution able to combine flows (of possibly different types)?
4.	**Support for multiparty connections**
4.1	Does the multimedia stream binding solution support point-to-multipoint connections?
4.2	Does the multimedia stream binding solution support multipoint-to-multipoint connections?
5.	**Support for stream synchronisation**
5.1	Does the multimedia stream binding solution support real-time flow synchronisation?
5.2	Does the multimedia stream binding solution provide methods to specify arbitrary synchronisation actions at run-time?
5.3	Does the multimedia stream binding solution provide synchronisation support for multiparty connections?

Table 10: Requirements of a multimedia stream binding solution

5.8 Summary

In this Chapter, a reference model is developed for the analysis and comparison of multimedia stream binding solutions. The reference model uses the ODP-RM Information and Computational Viewpoints as starting points for the analyses. Models for these viewpoints can be made by using UML diagrams.

In these diagrams, a number of reference points, which address the interesting issues of a solution (these issues are addressed by the Information and Computational Viewpoints) can be pointed out. These reference points are then used to make analyses of the solution from the Information and Computational Viewpoints. These analyses are then used to fill in an extensive checklist, which treats all requirements a multimedia stream binding solutions should meet. This strength/weakness analysis can be used to compare a solution with other solutions.

In the next Chapter the reference model is applied to a number of multimedia stream binding solutions. Using the resulting strength/weakness analyses, these solutions are compared and conclusions are drawn from this comparison.

6 Analysis and Comparison of Multimedia Stream Binding Solutions

6.1 Introduction

In this chapter, analyses and comparisons are made of the TINA Network Resource Architecture, OMG Control and Management of A/V Streams and ITU-T H.323 standards which are discussed in Chapter 4. The analyses are made by using the reference model developed in Chapter 5. These analyses are then used in validating the 'hybrid control binding object' and 'hybrid standards binding object' concepts, described in Chapter 3.

As already mentioned in Chapter 4, these three standards are selected because they differ in both design paradigm (object-centered or protocol-centered) and responsibility of control approach (central responsibility or distributed responsibility) so they represent heterogeneous solutions. These differences are shown in Table 11:

	object-centered	**protocol-centered**
central responsibility	TINA NRA	
	OMG A/V Streams	
distributed or delegated responsibility		ITU-T H.323

Table 11: Design Paradigms and Control Approaches of Standards

First, the strength/weakness analyses made of these standards by using the reference model from Chapter 5 are given. Then, this Chapter focuses on how these standards can co-operate through federation and integration, to validate the 'hybrid standards binding object'-concept. After this, the 'hybrid responsibility binding object'-concept is validated, by discussing the different control approaches.

6.2 Strength/Weakness Analyses

In this section it is described how the standards meet the items from the checklist given in Section 5.7. The other parts of the strength/weakness analyses (a short description of each standard and the analysis of each standard) can be found in Chapter 4 and a separate Research Note describing the analyses.

6.2.1 OMG Control and Management of A/V Streams

1.1 *Does the multimedia stream binding solution support all important protocols and standards?*
The supported protocols and standards are identified in the flow specification grammar. They are: TCP, UDP, AAL5, AAL3-4, AAL1, RTP/UDP, RTP/AAL5 and IPX. Media types (codecs) are described using MIME-types or IDL flow types. An unspecified type is also available. The OMG plans to register the protocols, standards and media types to keep the syntax consistent.

1.2 *Does the multimedia stream binding solution accept any incoming flow, independent of its type?*
Incoming flows are accepted by a Stream/Flow-endpoint, if the type of the flow is

supported by that endpoint, and, when using the full profile, if the endpoint is a *consumer* endpoint, Because the OMG Spec. make extensive use of data structures for QoS management and configuration, it is important that the originator also supports these data types.

1.3 *Is the multimedia stream binding solution able to bind to any other interface?*
When the Stream/Flow-endpoints are compatible, then a binding with that interface can be established, provided that the interface is able to communicate with the OMG Spec. objects.

2.1 *Does the multimedia stream binding solution provide methods to avoid jitter?*
Yes: this is regulated through QoS parameters. Realisation of this QoS parameter is dependent on other QoS and configuration parameters, like the media types and transport and control protocols used, the bandwidth available, the type of service, etc.

2.2 *Does the multimedia stream binding solution support bandwidth reservation and/or other resource reservation?*
Yes: this is regulated through QoS parameters. Special QoS parameters are the ServiceType-parameter, which can have the values 'Best Effort', 'Guaranteed' and 'Predicted', and the ErrorFree-parameter which determines wether the connection should be error free. Bandwidth and peak bandwidth can be specified through desired bandwidth and lower and upper bounds. Also, the maximum acceptable token rate, delay and jitter can be specified.
Multimedia device resources like cameras and microphones are reserved when creating a new stream endpoint. When no resources are available, then the stream endpoint can not be created, and the stream binding is not established.
All bandwidth reservations are determined by the QoS and configuration data types.

2.3 *Does the multimedia stream binding solution provide negotiation of Quality of Service?*
Yes: A specialised information object (streamQoS) is defined to hold all QoS data. QoS negotiation is initiated by the StreamCtrl-object, and performed by the VDev-objects located at each endpoint. In multipoint sessions, an MCastConfigIf-object is used to multicast the data from the sending party to the receiving parties.
When a requested QoS can not be fulfilled, the binding is not established, and an exception is thrown to the program that requested the binding

2.4 *Does the multimedia stream binding solution provide the change of the Quality of Service provided?*
Yes: changing the QoS is supported through explicit operations. Change of QoS is carried out by renegotiating the QoS with the other parties. If the requested change of QoS can not be established, then the connection is ended.
In the specification it is observed that today's protocols generally do not support QoS changes. To overcome this it is suggested to bring down a flow connection and to establish a new connection with a new QoS.

2.5 *Does the multimedia stream binding solution provide dynamic Quality of Service management, such as monitoring, maintenance, policing and renegotiation of the quality of service requested?*
In this issue the ServiceType-parameter plays an important role in specifying how strict QoS requirements should be fulfilled. When this parameter is set to 'Guaranteed' then the requested QoS *must* be met. When this parameter is set to 'Best Effort' or 'Predicted' the actual QoS may vary more or less. In order to support this (mandatory) parameter, there has to be some kind of dynamic QoS management facility, but in the OMG spec. such a facility is not described.
In general, the OMG spec. defines a means to specify a Quality of Service, but the realisation is left to the implementations and/or the used transport and control protocols.

3.1 *Is the multimedia stream binding solution able to bind to multiple flows?*
Yes: when using the full profile separate flows can be controlled. A stream endpoint can then bind to multiple flows originating from and going to different peers simultaneously, because in the stream configuration information object holds the address of each flow.

3.2 *Is the multimedia stream binding solution able to combine flows (of possibly different types)?*
No: combining (or mixing) flows is not addressed by the OMG spec. This capability is left to the underlying communication architecture and transport and control protocols. For example, when using the H.320 standard to establish a connection, an MCU is needed to provide these facilities. When using other transport and control protocols, other facilities might be needed.

4.1 *Does the multimedia stream binding solution support point-to-multipoint connections?*
Yes: point-to-multipoint connections can be established by using IP-multicast or ATM-style multicast. Separate computational objects are available for supporting point-to-multipoint connections. These objects are used to broadcast messages and data from the sender to all receivers. Different QoS agreements with different receivers is only possible when using ATM-style multicast.

4.2 *Does the multimedia stream binding solution support multipoint-to-multipoint connections?*
There is no native support for multipoint-to-multipoint (or mp-t-mp for short), but a mp-t-mp connection can be realised by means of combining a number of point-to-multipoint connections, so that every party is both a sender (to all other parties) and a receiver (of all other parties). Because a StreamEndPoint-object can be both a sending and receiving party, only one StreamEndPoint-object is needed per party.
Depending on wether the used communication architecture and transport and control protocols offer native support for mp-t-mp connections, these facilities can be used in setting up a mp-t-mp connection.

5.1 *Does the multimedia stream binding solution support real-time flow synchronisation?*
No: flow synchronisation is left to the device endpoints, so the hardware devices (or the software controlling those devices) and the transport and control protocols have to support flow synchronisation. The Simple Flow Protocol (SFP) however includes a timestamp field which can be used for synchronisation issues.

5.2 *Does the multimedia stream binding solution provide methods to specify arbitrary synchronisation actions at run-time?*
No: flow synchronisation is left to the device endpoints.

5.3 *Does the multimedia stream binding solution provide synchronisation support for multiparty connections?*
No.

6.2.2 TINA Network Resource Architecture

1.1 *Does the multimedia stream binding solution support all important protocols and standards?*
The TINA NRA is designed as an open standard and does not describe which protocols and standards are supported. TINA NRA however requires solutions and standards that support guaranteed QoS reservations. The design is also influenced by ATM capabilities.

1.2 *Does the multimedia stream binding solution accept any incoming flow, independent of its type?*
A connection in the TINA NRA is entirely set up and controlled by a set of centralised

stream control and management objects. So there are no incoming flows received directly by endpoints; each flow is set up by the centralised stream control objects.

1.3 *Is the multimedia stream binding solution able to bind to any other interface?*
Bindings in the TINA NRA are completely set up and managed by a number of centralised stream control and management objects. Binding establishment is initiated by a CSM object. A binding is rejected when the requested Stream Flow Connections can not be created.

2.1 *Does the multimedia stream binding solution provide methods to avoid jitter?*
Yes; the TINA NRA requires guaranteed QoS reservation capabilities. Jitter is not explicitly mentioned as a QoS parameter in the TINA NRA specification, but it is commonly accepted as an important QoS parameter for multimedia applications, so it is to be supported by the TINA NRA.

2.2 *Does the multimedia stream binding solution support bandwidth reservation and/or other resource reservation?*
Yes; bandwidth reservation and other resource reservation is required by the TINA NRA. Resource reservation is described at the connection level by Logical Connection Graphs, and at the network level by Physical Connection Graphs.

2.3 *Does the multimedia stream binding solution provide negotiation of Quality of Service?*
No; QoS reservation is entirely managed by a centralised CSM, no QoS negotiation between endpoints is possible. A specific QoS can be requested to the CSM object when the connection is created, and the CSM object takes care of the reservation of resources to establish the connection. When the requested QoS can not be met, the connection is rejected by the CSM.

2.4 *Does the multimedia stream binding solution provide the change of the Quality of Service provided?*
Yes: on several interfaces of computational objects modify() operations are available, making it possible to change several properties of these objects, including the QoS.

2.5 *Does the multimedia stream binding solution provide dynamic Quality of Service management, such as monitoring, maintenance, policing and renegotiation of the quality of service requested?*
No; the TINA NRA is heavily based on ATM, which is capable of reserving guaranteed bandwidth. Guaranteed QoS can be specified when these capabilities are available, which makes it not necessary to provide these features. The TINA NRA however provides operations to modify QoS parameters (the modify() operations on several interfaces) and operations to retrieve information about the state of (part of) a connection (the getInfo() operations on several interfaces). See Issue 3 of the Analysis of the Computational Specification for these operations.

3.1 *Is the multimedia stream binding solution able to bind to multiple flows?*
Yes: each endpoint has one TCSM object, which can serve as source/sink for multiple flows, possibly originating from or received by more than one peer TCSM. Because the TINA NRA provides very extensive methods to manage separate flows, it is relatively easy to realise such connections (such connections can be described by using a standard LCG).

3.2 *Is the multimedia stream binding solution able to combine flows (of possibly different types)?*
No; combining flows is not explicitly supported by the TINA NRA, because a communication session is described by a logical connection graph, which contains a rule that each leaf endpoint (which receives a flow) can be connected to only one root endpoint (which sends a flow). However, devices like MCUs can be modeled by creating a device that holds more leaf endpoints which receive the video flows, and one root endpoint which sends the mixed video flow.

4.1 *Does the multimedia stream binding solution support point-to-multipoint connections?*
Yes: the TINA NRA is explicitly designed to support point-to-multipoint connections. Such connections can be directly modeled in an LCG because multiple leaf flow endpoints can be connected to one root endpoint by using the concept of Stream Flow Connection Branches.

4.2 *Does the multimedia stream binding solution support multipoint-to-multipoint connections?*
The TINA NRA does not provide modelling concepts for multipoint-to-multipoint (mp-t-mp) connections, but an mp-t-mp connection can be realised by means of combining a number of point-to-multipoint connections, so that every endpoint holds both root and leaf flow endpoints.

5.1 *Does the multimedia stream binding solution support real-time flow synchronisation?*
Yes: lip synchronisation can be specified by a boolean parameter. The capabilities and implementation of this feature are not found.

5.2 *Does the multimedia stream binding solution provide methods to specify arbitrary synchronisation actions at run-time?*
Only lip synchronisation is provided, this feature can be turned on and off on arbitrary moments. (see 5.1).

5.3 *Does the multimedia stream binding solution provide synchronisation support for multiparty connections?*
No, lip synchronisation is only provided for streams originating from the same endpoint (see 5.1).

6.2.3 ITU-T H.323

1.1 Does the multimedia stream binding solution support all important protocols and standards? Video codecs supported by the H.323 standard are: H.261 and H.263. Audio codecs supported by the H.323 standards are: G.711, G.722, G.723, G.728, G,729. A H.323 terminal has to support the G711 codec; support for other audio codecs and video codecs is optional. An H.323 terminal may support other codecs as well. For transport and control, the H.323 standard uses TCP for the establishment of reliable connections for control, and RTP on top of UDP to establish unreliable connections for streams. The IP protocol is used for data transport.

1.2 Does the multimedia stream binding solution accept any incoming flow, independent of its type?
Incoming flows are accepted if the type of the flow is supported by the endpoint. Connections from other network types can be converted by a Gateway.

1.3 Is the multimedia stream binding solution able to bind to any other interface?
An H.323 terminal can establish a connection with another H.323 terminal, and with other terminal types connected to other networks, by using a Gateway. 'Other' terminal types that are supported are V.70, H.324, speech, H.322, H.320 and H.321 terminals.

2.1 Does the multimedia stream binding solution provide methods to avoid jitter?
The H.245 Control component provides a variable to specify the maximum amount of jitter to be allowed (jitterindication) for each logical channel. Jitter can be controlled by the Receive path delay component, which can add delays to streams to reduce the effects of jitter.

2.2 Does the multimedia stream binding solution support bandwidth reservation and/or other resource reservation?
Bandwidth allocation can be managed by a Gatekeeper. The H.245 Control component has very extensive options to specify the configuration of audio and video codecs (in terms of maximum bandwidth used, frames per second, sample rate, etc.), but hard

guarantees for resource reservations cannot be given. There is also a parameter to specify the maximum amount of skew.

2.3 Does the multimedia stream binding solution provide negotiation of Quality of Service?
There is no explicit QoS negotiation procedure, but terminals exchange information about their capabilities, and the used configuration may be re-negotiated at any time, for example to meet the maximum bandwidth specified by the Gatekeeper.

2.4 Does the multimedia stream binding solution provide the change of the Quality of Service provided?
See above; the used configuration can be re-negotiated at any time.

2.5 Does the multimedia stream binding solution provide dynamic Quality of Service management, such a monitoring, maintenance, policing and renegotiation of the QoS requested?
See above; no specific mechanisms for dynamic QoS management are used, but bandwidth use can be monitored by a Gatekeeper (if one is used), and renegotiation of the used configuration can be carried out at any time. Also, the parameters for maximum jitter and maximum skew can be monitored.

3.1 Is the multimedia stream binding solution able to bind to multiple flows?
Yes; an H.323 terminal can have multiple open connections at the same time, when this is supported by the used hardware.

3.2 Is the multimedia stream binding solution able to combine flows (of possibly different types)?
Yes; but only when an MP is present which is able to mix audio, data or video flows. Flows of different types can be mixed when a Gateway is present which is able to convert these flows.

4.1 Does the multimedia stream binding solution support point-to-multipoint connections?
Yes; point-to-multipoint connections are supported when an MCU is present on the network, and the MCU contains and MP which is able to process the audio, data and video streams.

4.2 Does the multimedia stream binding solution support multipoint-to-multipoint connections?
Yes; multipoint-to-multipoint connections are supported when an MCU is present on the network, and the MCU contains and MP which is able to process the audio, data and video streams.

5.1 Does the multimedia stream binding solution support real-time flow synchronisation?
Yes; a specialised component, the Receive path delay component takes care of flow synchronisation. The audio and video flows contain time tags which are used by the Receive path delay component to add delays in order to preserve the inter-stream synchronisation.

5.2 Does the multimedia stream binding solution provide methods to specify arbitrary synchronisation actions at run-time?
Yes; the Receive path delay component is used for this purpose (see the item above).

5.3 Does the multimedia stream binding solution provide synchronisation support for multiparty connections?
Yes; this can be accomplished by MCUs and Gateways, which can alter the time tags associated with media flows, and by the Receive path delay component, which interprets these time tags to synchronise the flows.

The results of these strength/weakness analyses are summarised in the following table:

	OMG A/V Streams	TINA NRA	ITU-T H.323
1. Support for different protocols			
1.1 various types	+	+/-	+/-
1.2 determine compatible interfaces	+	+	+/-
1.3 bind to other interface types	+	+	+
2. Support for QoS management			
2.1 control delay jitter	+	+	+
2.2 resource reservation	+	+	+/-
2.3 QoS negotiation	+/-	-	+/-
2.4 change of QoS	+	+	+
2.4 dynamic QoS management	+/-	-	+/-
3. Support for multiple flow connections			
3.1 multiple flows	+	+	+
3.2 combine flows	+	+	+/-
4. Support for multiparty connections			
4.1 point-to-multipoint connections	+	+	+/-
4.2 multipoint-to-multipoint connections	+/-	+	+/-
5. Support for stream synchronisation			
5.1 real-time stream synchronisation	-	+	+
5.2 specify arbitrary synchronisation actions	-	+/-	+
5.3 synchronisation support for multiparty connections	-	-	+

+ = yes, +/- = yes, but with restrictions, - = no

Table 12: Summary of Strength/Weakness Analyses

6.3 Validation of Hybrid Standards

As already discussed in Section 3.7.2, the hybrid standards multimedia binding object concept can be used to establish multimedia bindings between endpoints supporting heterogeneous standards (designed using the object-centered and protocol-centered paradigms). In this Section the hybrid standards multimedia binding object concept is validated by discussing a number of co-operation scenarios between the standards analysed by the reference model.

Co-operation can take place between standards residing at different endpoints, to establish a multimedia binding between those endpoints, but co-operation can also take place between standards residing in the same endpoint. In this case co-operation takes place between 'high-level' and 'low-level' standards.

Table 13 shows the design paradigms used by each standard in each phase of the binding (O = object-centered, P = protocol-centered):

	OMG A/V Streams	TINA NRA	ITU-T H.323
1. set-up of a signalling channel	O	O	P
2. configuration negotiation	O	O	P
3. connection set-up	O/P	O	P
4. connection	O/P	O	P
5. renegotiation	O/P	O	P
6. disconnect	O/P	O	P

Table 13: Design paradigms used by multimedia stream binding standards

In this table the first two phases of the OMG A/V Streams standard use the object-centered concept; the concept used for the last two steps depends on the standard

selected to set up the actual connection. This standard is selected in the configuration negotiation phase, and can be object-centered or protocol-centered. This co-operation between the OMG A/V Streams standard and another standard can exist via federation or integration.

The following co-operation scenarios are discussed in this Section:

1. Federation and integration of the OMG A/V Streams and ITU-T H.323 standards, where the OMG A/V Streams standard is used as a 'high-level' standard, using the H.323 standard as a 'low-level' standard.
2. Federation and integration of the OMG A/V Streams and TINA NRA standards, where the standards reside at different endpoints
3. Federation and integration of the TINA NRA and ITU-T H.323 standards, where the TINA NRA standard is used as a 'high-level' standard, using the H.323 standards as a 'low-level' standard

These scenarios treat all different co-operation combinations (between object-centered and protocol-centered, and central responsibility and distributed responsibility standards).

6.3.1 Federation and Integration of the OMG A/V Streams and ITU-T H.323 standards

The Federation Case

In this co-operation scenario the OMG A/V Streams standard is used as a 'high-level' standard, using the H.323 standard as a 'low-level' standard (see Figure 6-1).

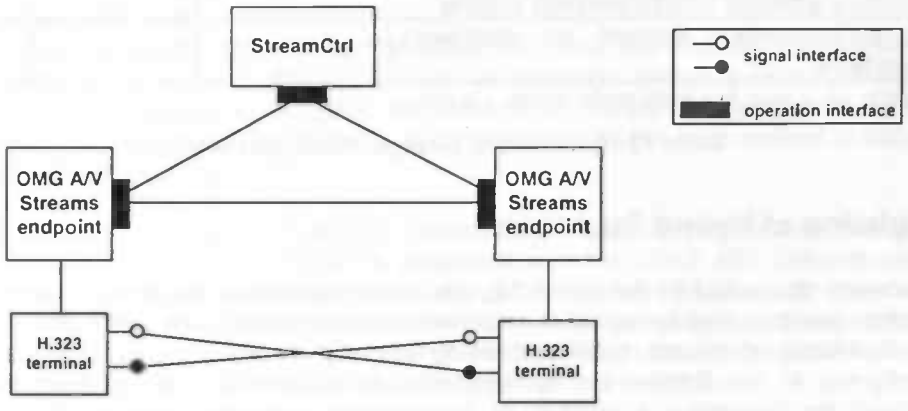


Figure 6-1: Federation between OMG A/V Streams and ITU-T H.323 terminals

In this Figure the H.323 terminals are directly connected to the OMG A/V Streams endpoints. This is done because these components are usually located at the same endpoint. The OMG A/V Streams endpoints communicate through operation interfaces, while the H.323 terminals communicate through signal interfaces. The H.323 implementation is usually accessible for the OMG A/V Streams endpoint implementation through an Application Programmer's Interface (API).

When using federation in a scenario as described here (use high-level standards to control low-level standards), the following connection phase sequence will often occur:

1. set-up of a signalling channel
2. configuration negotiation
3. connection set-up
    - 3.1 set-up of a signalling channel
    - 3.2 configuration negotiation
    - 3.3 connection set-up
4. connection
5. renegotiation
6. disconnect

From the connection set-up phase, the operation sequence is 'delegated' to the low-level standard to establish the actual connection (this is after the OMG A/V Streams request_connection()-operation has returned). The consequence of this scenario is that some tasks, like configuration negotiation, are carried out more than once. This is a trade-

off against the flexibilities offered by federation. It is possible to adapt phases of for example the high-level solution to remove overlapping parts (e.g. the configuration negotiation phase of the OMG A/V Streams standard could be adapted to leave QoS negotiation to H.323), but this requires special adaptations for each standard which federates as a low-level standard with the OMG A/V Streams standard.

In the OMG A/V Streams / ITU-T H.323 case, these adaptations are not very large because of the flexible configuration negotiation capabilities of the OMG A/V Streams by using a flow specification grammar. This grammar allows for example specification of the direction, flow protocol and address of each flow, which is sufficient information for doing a connection request to the lower-level H.323 standard.

An important conclusion which can be drawn here is that the OMG Control and Management of A/V Streams standard is very suitable for high-level binding management. It provides functionality for negotiating about which 'low-level' standard to use to establish the actual connection.

The Integration Case
When integrating the OMG A/V Streams and H.323 standards, the endpoint implementations consist of a combined OMG A/V Streams endpoint / H.323 terminal implementation. A possible application for this approach could be an OMG A/V Streams endpoint which is optimised for H.323 connections.

In this case, the preferred scenario is to perform the configuration negotiation and connection set-up phases by the OMG A/V Streams parts of the endpoints, and the connection phase by the H.323 parts of the endpoints. In this way the advantages offered by the OMG A/V Streams standard are best utilised (usage of a DPE, flexible configuration negotiation by using a flow specification grammar) to set up and manage an H.323 connection.

A disadvantage of this solution is that the OMG A/V Streams standard (especially the flow specification grammar and configuration negotiation parts) have to be adapted to support H.323-specific capabilities.

6.3.2 Federation and Integration of the OMG A/V Streams and TINA NRA Standards

The Federation Case
When a binding is set up between endpoints supporting the OMG A/V Streams standard and endpoints supporting the TINA NRA standard, this binding is controlled by the TINA NRA centralised objects (the CSM, CC, and other objects). Such a binding can not be controlled by the OMG A/V Streams centralised object (the StreamCtrl object) because this object delegates tasks (like configuration negotiation) to the endpoints, and the TINA NRA endpoints do not provide functionality for these tasks.

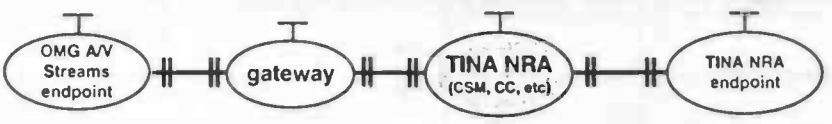


Figure 6-2: Binding between OMG A/V Streams and TINA NRA endpoints

When such a binding is established, a gateway object is needed to convert from TINA NRA object requests to OMG A/V Streams object requests. Both standards are object-centered, so no translation of object requests and PDUs is necessary.

The first problem is the conversion of the information objects of both standards. The TINA NRA uses connection graphs to describe flows, while the OMG A/V Streams standard uses a flow specification grammar and a QoS specification for this purpose. This conversion has to be provided by the gateway.

The second problem is that the configuration negotiation procedures of the OMG A/V Streams endpoint have to be disabled, because the configuration is determined by the

TINA NRA centralised objects. A proposed solution is to provide the gateway with StreamEndPoint interfaces, which 'negotiate' with the StreamEndPoint interface of the OMG A/V Streams endpoint. This way, configuration negotiation is redirected via the gateway. This 'negotiation procedure' can be used to pass the configuration determined by the TINA NRA centralised objects to the OMG A/V Streams endpoint. The same method can also be used for the configuration renegotiation phase.

In the connection set-up phase, the actual stream connection is set-up between the endpoints. In the TINA NRA this step consists of the CC, FCC, TM and TCM objects setting up network paths for the flow connections. In the OMG A/V Streams standard a connection is set-up by calling the connect()-operation on one of the StreamEndPoint-interfaces, which then calls the request_connection()-operation on each other StreamEndPoint-interface to retrieve the connection addresses of those interfaces. To keep control centralised, the request-connection()-calls have to be redirected. This could be done the same way as in the previous step: when a connect()-request is issued to a StreamEndPoint-interface, it holds a parameter specifying the address(es) of the peer StreamEndPoint-interface(s). These addresses could used to redirect the calls to the gateway.

From this scenario it can be concluded that a gateway component for the establishment of hybrid multimedia bindings needs some 'intelligence' next to conversion of operations and information objects. This is due to the different designs and behaviour of components in especially the configuration negotiation and connection set-up phases.

The Integration Case
When using integration, this means that either the OMG A/V Streams components have to be extended to support the TINA NRA, or opposite. Because in the configuration discussed in the previous paragraph the TINA NRA components are used in both the endpoints and centralised components, it is believed to be the best method to extend the OMG A/V Streams components.

In this case the interface conversion functionality (which is provided by the gateway component in the federation case) is integrated with the OMG A/V Streams endpoint. Because of this integration, the implementation of such an endpoint can be more efficient because the OMG A/V Streams implementation can be specifically adapted to the TINA NRA interfaces (e.g. the internal information representation can be adapted to the TINA NRA information representation, so no extra conversion is needed).

6.3.3 Federation and integration of the TINA NRA and ITU-T H.323 standards

The Federation Case
In this scenario the TINA NRA standard is used to set up network paths, and for high-level binding control and management, and the H.323 standard is used for end-to-end communication, using these network paths (Figure 6-3 shows this configuration).

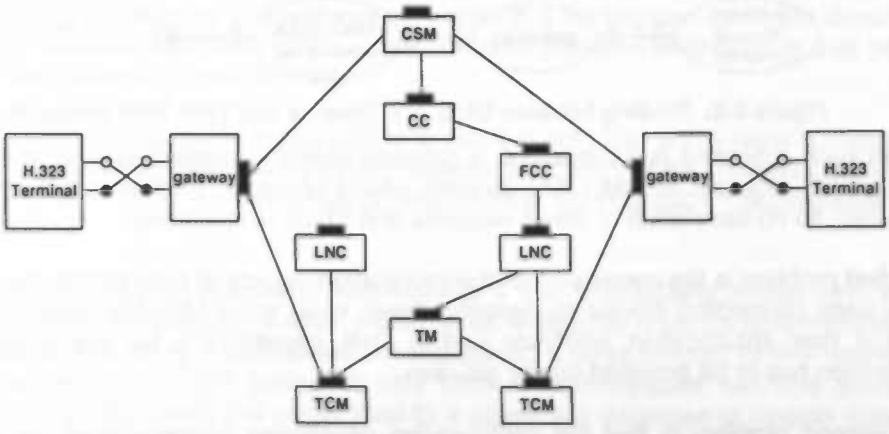


Figure 6-3: Federation between TINA NRA objects and H.323 terminals

In this configuration a gateway is used to translate TINA NRA operations to H.323 messages. As is shown in the Figure, the gateways only communicate with the CSM and TCM objects. This is sufficient because the gateways in fact offer TCSM and TLA interfaces to the centralised objects, which only communicate with the CSM and the TCM objects.

A binding is set up by the CSM first requesting capability information from the endpoints (via the gateways). The gateways have to transform this information by creating TINA NRA connection graphs from message structures describing properties of the H.323 terminals. Knowing this information, the CSM requests the CC to set up a network path between the endpoints (see Section 4.3.6). After this, the CSM passes the selected configuration to the H.323 terminals via the gateways. The gateways have to act as H.323 terminals in a configuration negotiation, to configure the H.323 terminals as requested.

The last phases in the binding process are the connection set-up and connection phases. The TINA NRA objects have already set up a network path between the gateways. This network path has to be used by the H.323 terminals to establish the connection. This can be accomplished by the gateway of the initiating party acting as peer H.323 endpoint (so the real H.323 endpoint establishes a connection with the gateway), and then using the TINA NRA network path to transfer this connection to the peer gateway, which connects to the peer H.323 terminal by using standard H.323 signalling.

Like in the OMG A/V Streams / TINA NRA federation scenario, the gateway components used need more functionality than just conversion of information objects and operation requests. They also need functionality to 'mislead' the H.323 terminals in order to achieve configuration negotiation and connection establishment.

The advantage of this scenario is that centralised responsibility can be preserved, while H.323 connections are still possible. The configuration is determined by the TINA NRA centralised objects and has to be transferred to the H.323 endpoints. The gateway objects are responsible for this. A drawback of this approach is that the TINA NRA objects need to know the H.323 configuration properties. Also, the network connection set up by the TINA NRA needs to support IP, because H.323 is an IP-oriented standard.

6.4 Validation of Hybrid Responsibility

As mentioned in Section 3.6, establishment of a stream binding using a multimedia binding object consists of six phases: set-up of a signalling channel, the configuration negotiation phase, the connection set-up phase, the connection phase, a renegotiation phase and a disconnect phase. For each of the three standards it can be classified whether a phase is accomplished using central responsibility or distributed responsibility. This leads to the following table (C = Central responsibility, D = distributed responsibility):

	OMG A/V Streams	TINA NRA	ITU-T H.323
1. set-up of a signalling channel	C	C	D
2. configuration negotiation	C/D	C	D
3. connection set-up	C/D	C	D
4. connection	D	C	D
5. renegotiation	C	C	D
6. disconnect	C/D	C	D

Table 14: Control approaches for multimedia stream binding standards

This table shows that the first step of stream binding establishment in the OMG A/V Streams standard is accomplished by using central responsibility. The second phase is initiated by the centralised object (thus by responsibility), the actual negotiation is accomplished by distributed responsibility. In the last steps the approach used depends on what standard is used to establish the actual connection. During configuration negotiation one or more standards are selected which will establish the actual connection.

These standards can use either central responsibility or distributed responsibility, or can also use a hybrid responsibility approach as described here.

From this it can be concluded that the OMG A/V Streams standard uses hybrid responsibility. In the remainder of this Section this standard is compared with the TINA NRA and ITU-T H.323 standards to investigate the advantages and disadvantages of the hybrid responsibility concept with respect to central responsibility control and distributed responsibility.

When comparing this Table with Table 11, it is observed that both tables are the same, except for the column titles. From this observation it can be concluded that standards which follow the object-centered paradigm mostly use centrally managed control, and that standards which follow the protocol-centered paradigm mostly use delegated control.

Comparing Central Responsibility and Hybrid Responsibility
Standards using the central responsibility concept (like TINA NRA) are responsible for every aspect of a multimedia stream connection, from high level binding configuration to setting up a communication path by controlling switches, routers and other network elements. In a hybrid standard, responsibility is delegated to the endpoints at a certain phase in setting up a connection.

The advantage of central responsibility is that there is a total overview of usage of network resources, so control and management of network resources is relatively easy. Disadvantages of central responsibility are that a large amount of information has to be managed, especially in inter-network connections, and that it is very difficult to co-operate with other (endpoint managed) standards, because these likely have to be adapted to support central responsibility. Also, network elements which are needed in the binding have to be adapted to support this.

Standards using hybrid responsibility, on the other side, profit from the control and management capabilities of centralised responsibility, but they do not suffer the obligation to be responsible for every aspect of the connection. This makes it much easier to use other technologies for low-level binding establishment, because they are responsible for configuration and connection set-up.

An important issue in this comparison is when responsibility should be delegated to the endpoints. In the OMG A/V Streams standard, this is done in the configuration negotiation phase. Nevertheless, central responsibility is held by the (centralised) StreamCtrl object. This approach works very well, because information about capabilities of endpoints is stored in these endpoints, and only the used configuration is passed to the other objects. In this way the centralised objects do not have to be aware of all configuration parameters used by the endpoints.

Comparing Distributed Responsibility and Hybrid Responsibility
Standards using the distributed responsibility concept (like ITU-T H.323) do not use any centralised control objects. The endpoints involved in a binding communicate directly with each other. The advantage of direct communication between endpoints is that no third party is needed to establish a binding.

A disadvantage of distributed responsibility is that it is not clear which party is responsible for the whole binding. When using hybrid responsibility, there is a centralised party which can control the status of the binding, and initiate configuration modifications when necessary.

Another disadvantage of distributed responsibility is that in multiparty connections, a large communication overhead is needed because all endpoints have to communicate with each other. When using a centralised binding object, a large amount of this overhead is not necessary because the endpoints only have to communicate with one centralised object. In the H.323 case, this would lead to a significant decrease of update messages which are exchanged between the endpoints.

6.5 Summary

In this Chapter strength/weakness analyses of the TINA NRA, OMG Control and Management of A/V Streams and ITU-T H.323 standards are described. These were made by using the reference model developed in Chapter 5. These standards were selected because they differ in both design paradigm (object-centered or protocol-centered) and responsibility approach (central responsibility or distributed responsibility).

These standards are used to validate the 'hybrid standards' and 'hybrid responsibility' multimedia binding object concepts. It is concluded that the TINA NRA standard is designed using the object-centered paradigm and uses central responsibility, the OMG A/V Streams standard is designed using the hybrid standards paradigm and uses hybrid responsibility, and the ITU-T H.323 standard is designed using the protocol-centered paradigm and uses distributed responsibility.

To validate the hybrid standards concept, co-operation of combinations of these standards are investigated. Both co-operations by federation and integration are investigated. A conclusion which can be drawn from these investigations are that the OMG Control and Management of A/V Streams standards is best used for high-level binding control and that the H.323 standard is best used as a specific technology for videoconferencing over IP-networks.

Another conclusion is that co-operation through federation is a better solution than co-operation through integration. Advantages of federation are that co-operation with other standards is easier to achieve by using *gateways*. These gateways take care of conversion of information objects and operation calls. However, such a gateway generally needs more intelligence to anticipate to different behaviour of various solutions. Also, it is not necessary to have access to the implementation of the standards to be used. This is in contrast to when integrating standards, because to achieve a good integration the implementations of the standards involved need to be adapted.

An advantage of integration against federation is that the resulting implementations are usually more efficient, because co-operation between standards can be better optimised. Nevertheless, this usually does not counterbalance against the advantages offered by federation.

7 Validation of a Hybrid Solution: A Distributed Videoconferencing Application

7.1 Introduction

In this Chapter the design and implementation of a simple, point-to-point hybrid multimedia binding object is described, which validates the hybrid multimedia binding object concept. For this implementation, a simplified implementation of the OMG A/V Streams standard is used, co-operating through federation with the ITU-T H.320 standard, which provides videoconferencing facilities over ISDN networks. The design of the H.320 standard is essentially the same as the H.323 standard design. Communication between both standards is also possible (see Section 4.4 for more information).

This demonstrator was built as a part of the EURESCOM P715 project. The objective of this project is to build a services platform using middleware technologies based on TINA concepts, to demonstrate the capabilities of this architecture to provide value added services. The goal of this demonstrator was to add multimedia services to the services platform.

7.2 Design of a Hybrid Solution Using the OMG A/V Streams and H.320 Standards

The solution described in this Chapter is based on the OMG Control and Management of A/V Streams specification and uses CORBA as DPE. The architecture used is a simplified version of the OMG A/V Streams specification's 'light profile'. An overview of this architecture of the solution is shown in Figure 7-1.

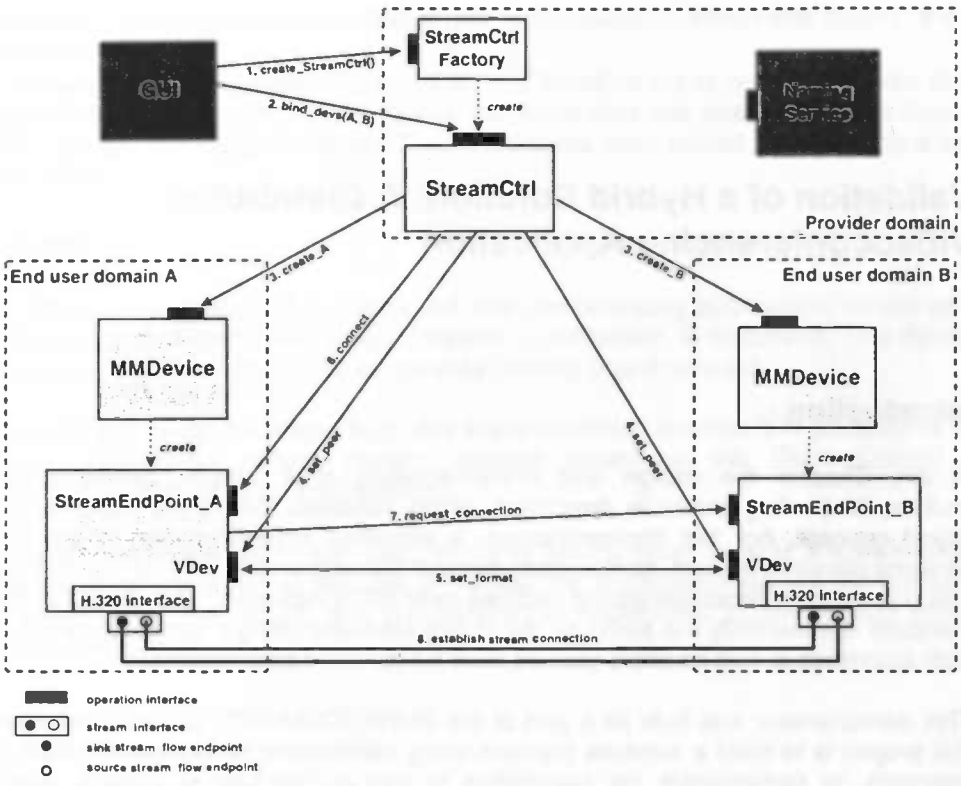


Figure 7-1: Hybrid Multimedia Binding Object Architecture (Computational Viewpoint)

The following components are shown in this Figure:

- *GUI* - this component lists the available endpoints between which a binding can be established. A user can select two endpoints and request a binding establishment between these endpoints
- *Naming Service* - this component implements the CORBA Naming Service (see [OMG, 1997])
- *StreamCtrl Factory* - creates StreamCtrl components
- *StreamCtrl* - sets up, controls and manages the binding
- *MMDevice* - manages endpoint devices; creates StreamEndPoints and VDevs as long as resources are available
- *StreamEndPoint* - abstracts from a stream endpoint. Establishes the actual connection
- *VDev* - abstracts from hardware devices. Carries out configuration negotiation by exchanging flow specification strings (in this implementation the flow specification strings are empty because it is already known that H.320 is used for the connection)
- *H.320 interface* - provides an interface to the H.320 standard

The Figure also shows End User domains and a Provider domain. This denotes the location of a component. The GUI is not located in one of these domains, because the GUI is implemented as a Java applet which runs in a web browser.
A binding is established by carrying out the following (simplified) operation sequence. The full operation sequence, which also describes the disconnection sequence, can be found in Appendix C.

1. The GUI issues the StreamCtrl-Factory to create a StreamCtrl object. The StreamCtrl object reference is returned to the GUI.
2. The IORs of the MMDevs of the calling and the called party are the parameters for the *bind_devs*-request to the StreamCtrl object.
3. (3a). The StreamCtrl does a *create*-request to the two MMDevs: a *create_A*-request to the calling MMDev, and a *create_B*-request to the called MMDev.
   (3b) The calling MMDev creates a StreamEndPoint_A object and a VDev object, the called MMDev creates a StreamEndPoint_B object and a VDev object. (5c) The references to these objects are returned to the StreamCtrl.

4. The StreamCtrl does a *set_peer*-request on the VDev objects to initiate the negotiation procedure between the VDevs to set up a connection with the requested specifications
5. The VDevs configure the requested connection parameters by calling the *operations set_format* and *set_dev_params* on the peer VDev. These operations check if all the flows originating from a VDev can be understood by the corresponding flow consumers on the peer VDev.
   Additional configurations can be accomplished by calling the operation *configure*.
6. The StreamCtrl issues a *connect*-request to the StreamEndPoint_A stream endpoint to establish a connection with desired specifications.
7. The StreamEndPoint_A object sets up the connection by doing a *request_connection*-request on the StreamEndPoint_B. This request returns the phonenumber of the peer endpoint.
8. The StreamEndPoint_A establishes a stream connection between the stream endpoints, by issuing the H.320 interface to dial the peer H.320 interface.

7.3 The Implementation

To validate a multimedia binding service in a heterogeneous environment, different platforms, programming languages and video hardware are used to implement the parts of the demonstrator.

The GUI is implemented as a Java applet and runs inside a Web browser. The StreamCtrl Factory and StreamCtrl components were implemented in Java, and run on a UNIX environment. The endpoints were implemented in C++ and run on Windows NT environments. For the communication between operational interfaces different CORBA implementations are used. Also, different types of video hardware are used on the endpoints.

For a detailed description of the platforms, programming languages, CORBA implementations and video hardware used, see Appendix C. This Appendix also gives a detailed description of the implementation of one of the endpoints.

7.4 Conclusions / Lessons Learned

A important conclusion which can be drawn from this experiment is that it is possible to build a distributed multimedia stream binding solution based on the OMG Control and Management of A/V Streams specification. The implementation is however very limited and should be extended to support other technologies like H.323. Another interesting feature to add is the capability to set up and manage multipoint connections.

Despite the usage of only a subset of the OMG A/V Streams specification, a number of unclarities of the specification were encountered during the design and implementation process:

- The location (End User domain or Provider domain) of components like StreamEndPoint and VDev is not clear
- It is not described in the specification which party is responsible for creating StreamCtrl objects. Therefore a StreamCtrl Factory object, which provides this facility, was added
- The working of the configuration negotiation procedure was not clear. According to the specification, configuration negotiation is carried out by the VDev objects of the endpoints, but it is not possible to return the new configuration to the StreamCtrl or StreamEndPoint objects

Other problems encountered during the implementation: (these problems are more related to the used CORBA implementation and developer's kit to control the videoconferencing hardware; for a detailed description of the used platforms, programming languages and video hardware, see Appendix C):

- The CORBA specification does not allow overloading of operations in derived classes. The Orbix 2.3 IDL compiler however, does allow this
- The Proshare Developer's kit, which implements the H.323 standard, often gets into an undetermined state when the endpoint implementation crashes or when trying to

establish a new connection too fast after a disconnection. When this occurs, the PC has to be rebooted

- Orb-interoperability (between Orbix v.2.3c and CoolOrb v.4.1) posed problems due to bugs in Orbix (communication between the Orbs was not possible due to incorrect IIOP address translation in Orbix). These problems were solved after running a patch from IONA.

7.5 Summary

In this Chapter the design of a distributed videoconferencing application, based on the OMG Control and Management of A/V Streams specification, is discussed. This design is implemented to validate the hybrid multimedia binding object concept.

Conclusions which can be drawn from this implementation are that it is possible to build distributed multimedia stream binding solution based on the OMG Control and Management of A/V Streams specification.

The specification however, is not clear in some points, most notable the configuration negotiation phase. Interesting features to extend the implementation are the addition of other technologies like H.323, and the addition of capabilities to set up and manage multipoint connections.

8 Conclusions & Recommendations

Conclusions:
A large number of de jure and de facto standards for the set up, control and management of multimedia connections are available, developed by both the telecommunications and information technology industry. A distinction can be made between standards designed according to the object-centered paradigm, and standards designed according to the protocol-centered paradigm.

The 'hybrid multimedia binding object'-concept is a good concept for capturing this convergence of the Telecommunications and IT sectors in the area of multimedia applications. The term 'hybrid' however, is too general and has to be refined into two terms: 'hybrid responsibility' and 'hybrid standards'.

Validation of the hybrid responsibility multimedia binding object concept concludes that hybrid responsibility is preferred above central responsibility and distributed respon-sibility. It provides the control and management advantages of central responsibility and easy adaptation to new technologies provided by distributed responsibility.

Validation of the hybrid standards multimedia binding object concept concludes that co-operation of standards through federation is a better approach than co-operation through integration. When using federation, in most cases *gateways* are needed to convert information objects and to translate object centered operation requests to protocol centered message calls. From the six phases which can be distinguished in a multimedia binding life-cycle, the 'connect-phase' always uses the protocol-centered approach. The reason for this is that this approach is more efficient (it is unnecessary overhead to 'pack' the multimedia data into objects and to 'unpack' them again at the destination).

The OMG Control and Management of A/V Streams specification is an example of a hybrid multimedia binding object, providing hybrid responsibility and support for hybrid standards. The standard is designed for high-level binding control and uses 'lower-level' standards (e.g. H.323) to establish the actual binding.

Recommendations:
In this thesis a rather theoretical validation of the hybrid multimedia binding object concept was carried out, resulting in a number of possible co-operation scenarios. It is recommended that practical implementations of these scenarios are made to validate the implementation issues of these scenarios.

A first implementation, based on the OMG A/V Streams specification and the ITU-T H.320 standard is made. This implementation is very proprietary and only implements a subset of the OMG A/V Streams specification. The implementation should therefore be rewritten to support the full specification, and should be extended to support more connection establishment standards (e.g. H.323) and multiparty connections. It is recommended to use Microsoft Netmeeting for such a new implementation, because this would make the implementation hardware independent. An Software Development Kit (SDK) of Netmeeting can be downloaded from Microsoft's web site.

It is interesting to do performance measurements (with respect to speed, efficiency, etc.) of a number of implementations, and to make comparisons between these implementations and (commercial) vendor implementations.

9 References

[AF] ADSL Forum, URL: http://www.adsl.com

[Blair, 1997] G, Blair, J.B. Stefani, *Open Distributed Processing and Multimedia*, Addison Wesley Longman Ltd., Harlow, 1998

[Booch, 1997] G. Booch, *Quality Software and the Unified Modeling Language*, Rational Rose Website, UML White Papers

[Chiarglione, 1996] L. Chiarglione, *MPEG and multimedia communications*
URL: http://www.cselt.stet.it/ufv/leonardo/paper/isce96.htm

[Chiarglione, 1997] L. Chiarglione, *The Challenge of Multimedia Standardization*, IEEE multimedia, pp. 79-83, April-June issue, 1997

[DAVIC CFP12] DAVIC's Twelfth Call for Proposals, *Generic Multimedia Contribution Systems and Components*, Milan, March 1998

[DAVIC CFP13] DAVIC's Thirteenth Call for Proposals, *IP Forwarding Mechanisms for the Control of IP Network Performance*, Milan, March 1998

[De Muijnck, 1997] J.H. de Muijnck *et al., Gateway-, multipoint- en retrievaldiensten gebaseerd op H.323 en streaming*, Report R&D-RA-97-964, KPN Research, 1997

[Ensor, 1997] J.R. Ensor, S.R. Ahuja, *Communication Middleware for Multi-Party Multimedia Applications*, Bell Labs Multimedia Communication Research Department, Holmdel, New Yersey, 1997
URL: http://www.lucent.com/ideas2/perspectives/bltj/winter_97/pdf/paper06.pdf

[Ferguson, 1998] P. Ferguson, G. Huston, '*Quality of Service in the Internet: Fact, Fiction or Compromise?*', to be published at in the proceedings of INET '98, Geneva, July 1998
URL: http://www.employees.org:80/~ferguson/inet_qos.htm

[Fowler, 1997] M. Fowler, K. Scott, *UML Distilled*, Addison Wesley Longman Inc., 1997

[Gay, 1997] V. Gay, P.Leydekkers, *ODP-RM for Multimedia*, IEEE multimedia - standards, pp. 68-73, January-March issue, 1997

[GEA] Gigabit Ethernet Alliance, URL: http://www.gigabit-ethernet.com

[Grosky, 1997] W. Grosky, '*Pushing Streaming Video*', IEEE Multimedia p. 6, October-December issue, 1997

[ITU H245, 1996] *Control protocol for multimedia communication*, ITU-T Recommendation H.245

[Leydekkers, 1997] P. Leydekkers, *Multimedia Services in Open Distributed Telecommunications Environments*, PhD thesis, KPN Research, 1997

[Lu, 1996] G. Lu, *Communication and Computing for Distributed Multimedia Systems*, Artech House Inc., Norwood, 1996

[MCI1, 1997] *Relationship of the Unified Modelling Language to the Reference Model of Open Distributed Computing*, MCI Systemhouse Corporation, 1997
URL: http://enterprise.Systemhouse.MCI.com/UML-ODP

[MMCX] For more information, search the Lucent Web-site (URL: http://www.lucent.com/ideas2/ideas.html) for 'MMCX'.

[Muhlhauser, 1996] M. Muhlhauser, J. Gecsei, *Services, Frameworks and Paradigms for Distributed Multimedia Applications*, IEEE multimedia, pp. 48-61, Fall issue, 1996

[Netmeeting] Microsoft Netmeeting Homepage.
URL: http://www.microsoft.com/netmeeting

[Noorlander, 1998] N.E. Noorlander, *Analysis and Design of a DVB Interaction Channel*, KPN Research, dept. SAM, 1998

[OMG, 1997] Object Management Group, *'The Common Object Request Broker: Architecture and Specification, Revision 2.0'*, OMG document 97.2.25, 1997

[OMG A/V Streams] Object Management Group, *'CORBA Telecoms'*, Object Management Group Inc., 1997

[ODP-RM 1, 1995] ITU/ISO, Open Distributed Processing - Reference Model, *Part 1: Overview*, International Standard 10746-3, ITU-T Recommendation X.903, 1996

[ODP-RM 2, 1995] ITU/ISO, Open Distributed Processing - Reference Model, *Part 2: Foundations*, International Standard 10746-3, ITU-T Recommendation X.903, 1996

[ODP-RM 3, 1995] ITU/ISO, Open Distributed Processing - Reference Model, *Part 3: Architecture*, International Standard 10746-3, ITU-T Recommendation X.903, 1996

[ODP-RM 4, 1995] ITU/ISO, Open Distributed Processing - Reference Model, *Part 4: Architectural Semantics*, International Standard 10746-3, ITU-T Recommendation X.903, 1996

[Rose, 1997] C. Rose, *'Directories and LDAP - Universal Access to Directory Information'*, Netscape View Source, July 1997
URL: http://developer.netscape.com/tech/directory/index.html

[Schaphorst, 1996] R. Schaphorst, *Videoconferencing and Videotelephony: technology and standards*, Artech House Inc., Norwood, 1996

[Sinderen, 1997] M. van Sinderen, L.F. Pires, *Protocols versus Objects: Can Models for Telecommunications and Distributed Processing Coexist?*, Centre for Telematics and Information Technology, University of Twente, 1997

[Tanenbaum, 1988] A.S. Tanenbaum, *Computer Networks*, Prentice-Hall Inc., Englewood Cliffs, 1988

[TINA NRA, 1997] F. Steegmans et al., *TINA Network Resource Architecture v.3.0*, TINA Consortium, 1997

[UML1, 1997] *UML Notation Guide version 1.1*, Rational Software Corporation, 1997

URL: http://www.rational.com/uml

[Webopedia] *Webopedia - the number one online encyclopedia dedicated to computer technology*, Mecklermedia Corporation, 1998
URL: http://www.webopedia.com

[White, 1997] P.P. White, J. Crowcroft, '*The Integrated Services in the Internet: State of the Art*', Proceedings of the IEEE, pp. 1934-1946, December 1997

[Wolf, 1997] L.C. Wolf, C. Griwodz, R. Steinmetz, '*Multimedia Communication*', Proceedings of the IEEE, pp. 1915-1933, December 1997

Appendix A - The Unified Modelling Language (UML)

UML is a method for specifying, visualizing, and documenting the artifacts of an object-oriented system under development [Booch, 1997]. It is the unfication of the Booch, Objectory and OMT methods and it incorporates ideas from other methods as well. UML encompasses a number of diagrams that can be used to model and design a (possibly distributed) application. This Appendix provides only a short introduction to UML. For a more extensive coverage, see for example [Fowler, 1997].

UML provides the following modelling concepts to describe a system:

- Business process modelling
- Class and object modelling
- Component modelling
- Distribution and deployment modelling

Each concept is described using a set of UML diagrams. These diagrams are described in the following paragraphs.

Use-case Diagrams

Use-case diagrams are used to perform business process modelling. A use case diagram shows the system's use-cases and the actors interacting with them. Use-case diagrams are used for Business process modelling.

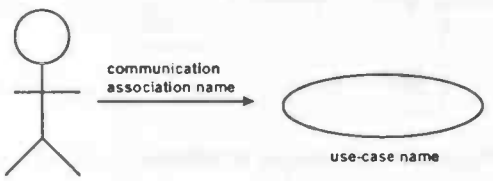


Figure 9-1: Use-case diagram

Class Diagrams

Class diagrams are used to show the important abstractions in a system and how they relate to each other. Thus, a class diagram represents the static sematics of a system. Figure 9-2 shows the elements and principles of a class diagram. Class Diagrams are used for Class and object modelling.

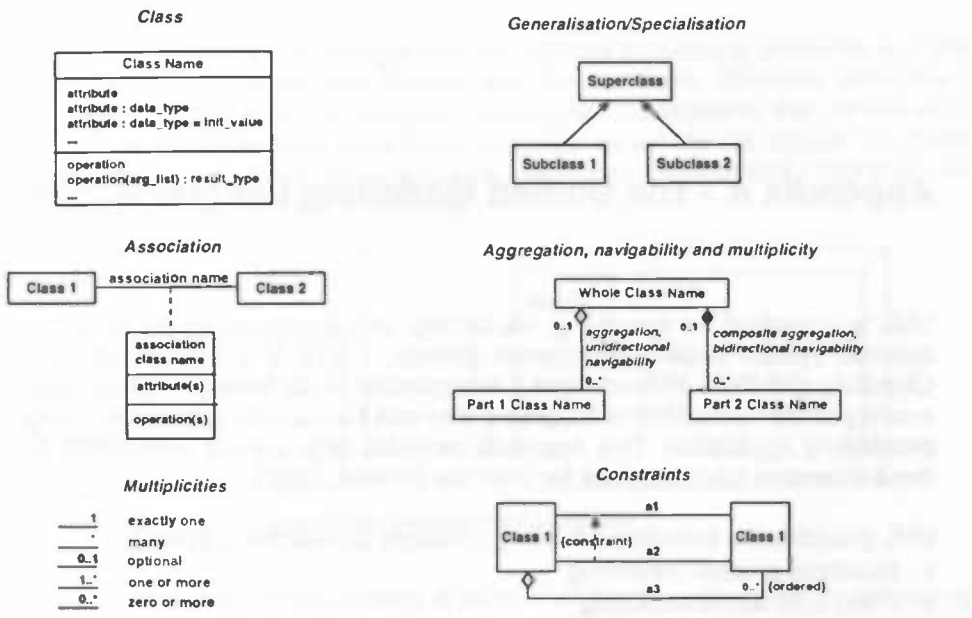


Figure 9-2: Elements and Principles of a Class Diagram

State-Transition Diagrams

State-transition diagrams are used to show the sequences of states that an object or an interaction goes through during its life in response to received stimuli, together with its responses and actions. Figure 9-3 shows the elements of a state-transition diagram. State-Transition Diagrams are used for class and object modelling.

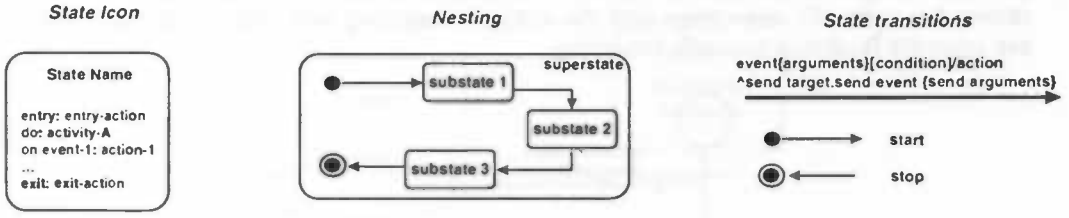


Figure 9-3: Elements of a State-Transition Diagram

Collaboration Diagrams

Collaboration diagrams are used to show interactions organized around the objects in the interaction and to each other. A collaboration diagram shows the relationship among the object roles. Collaboration Diagrams are used for class and object modelling.

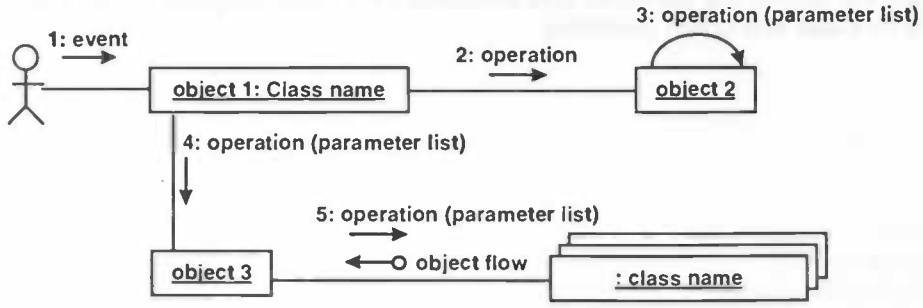


Figure 9-4: Collaboration Diagram

Activity Diagrams

Activity diagrams are a variation of state machines, which consist of action states, or activities, and transitions. A transition is triggered by completion of the actions in the source states. When more transitions are possible, a decision has to be make which one to take. The purpose of an activity diagram is to focus on flows driven by internal processing. Activity Diagrams are used for class and object modelling.

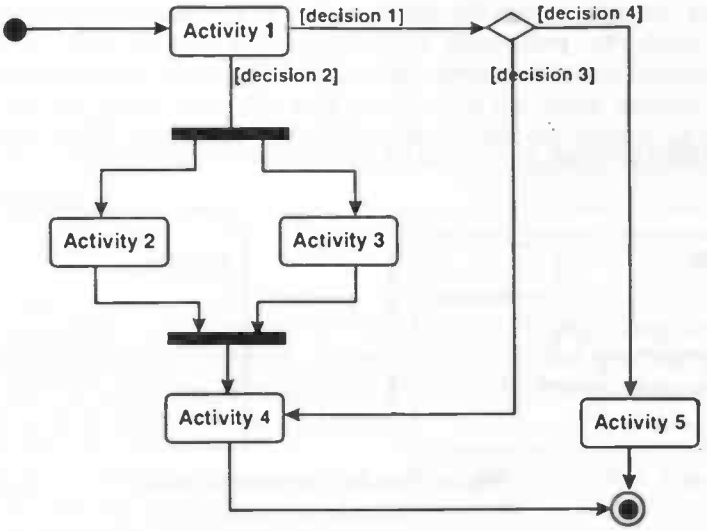


Figure 9-5: Activity Diagram

Sequence Diagrams

Sequence diagrams are used to show interactions between objects arranged in time sequence. It shows the objects participating in the interaction by their 'lifelines', and the messages that they exchange arranged in time sequence. Sequence Diagrams are used for class and object modelling.

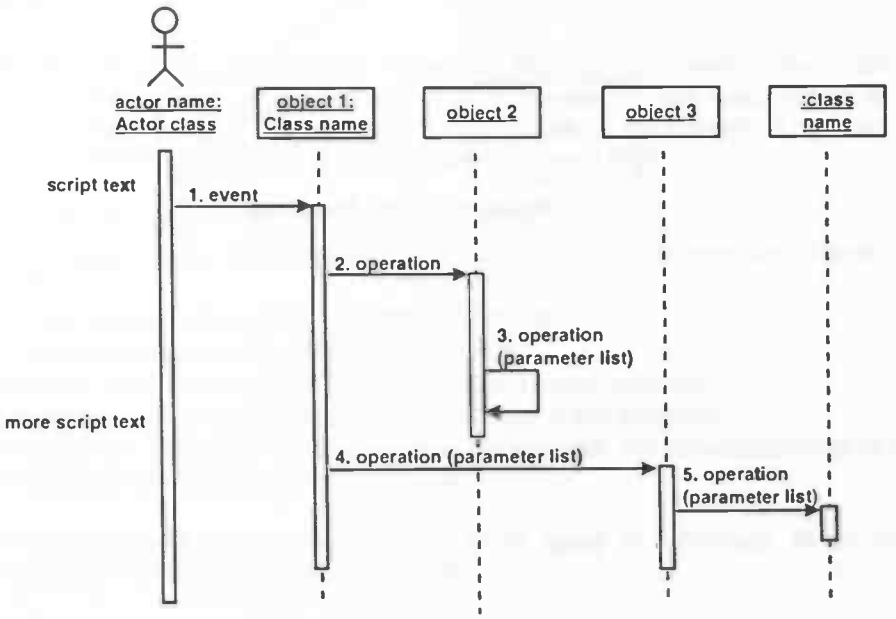


Figure 9-6: Sequence Diagram

Component Diagram

A component diagram shows the dependencies and relations between software components. Examples of components are source code components, binary code components and executable components. Component Diagrams are used for component modelling.

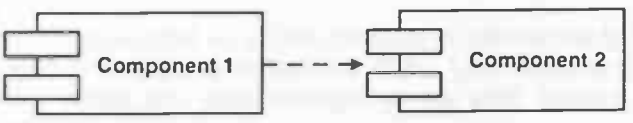


Figure 9-7: Component Diagram

Deployment Diagram

A deployment diagram shows the configuration of run-time processing elements and the software components, processes and objects that live on them. Software component instanse represent run-time manifestations of code units. Components that do not exist as run-time entities (because they have been compiled away) do not appear on these diagrams; they should be shown on component diagrams. Deployment Diagrams are used for distribution and deployment modelling.

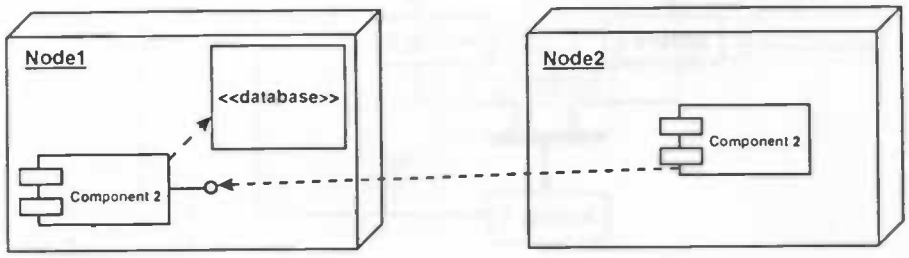


Figure 9-8: Deployment Diagram

Packages

A package is a special entity used to group a number of diagram elements. A package is denoted by a large rectangle with a small rectangle (a 'tab') in the upper left corner. Packages can be nested, and relations between packages can also be given.

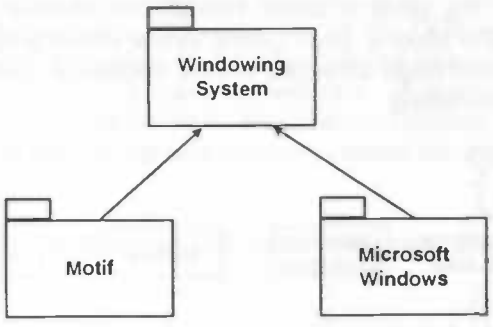


Figure 9-9: UML Packages

Appendix B - Relation Between ODP-RM Viewpoints and UML Concepts

It is expected that UML is going to be the standard for OOA&D languages. There are already tools available that support UML. Because UML fits well into the software design process, it can be expected that a large number of OO-software designers are going to use UML-based tools during the software design.

The UML concepts are also closely related to the ODP-RM viewpoints. Therefore they can be very well used for the analysis of the viewpoints. The next sections show how the concepts used to create the ODP-RM Viewpoint specifications can be mapped onto UML concepts.

Because in this thesis only the Information Viewpoint and the Computational Viewpoint are needed, only their relationship with UML concepts are treated. For a complete overview of the relationship of UML to ODP-RM, see [MCI1, 1997].

Relations between Information Viewpoint concepts and UML concepts

A specification from the Information Viewpoint focuses on viewing the information semantics and information processing activities in a system. This specification is called the information specification, or the 'information model' of the system. The information model has to answer the following questions [Leydekkers, 1997]:

- What are the information objects of the system?
- What manipulations/processing can be performed on the information objects of the system?
- What is the relationship between information objects?
- What are the attributes of the information objects?
- What are the rules and constraints for information manipulations?
- What are the sources, sinks, and information flows in the system?
- What semantics would be associated by a human with the information that is stored and exchanged between information objects?

The information model is modeled by using three types of schemata, which express different kinds of relationships between objects:

- the *invariant schema*: expresses the relationships between information objects, which must always be true, for all valid behaviour of the system
- the *static schema*: expresses assertions which must be true at a single point in time.
- the *dynamic schema*: specifies how the information can evolve as the system operates

The following UML correspondence rules apply to these schemata:

Invariant Schema
An *invariant schema* is represented as a UML package of references to UML objects. The UML constraints, together with the specifications of the UML classifiers of these objects, constrain the possible states and state changes of the UML objects. (A classifier can represent a class, datatype, or interface).

An invariant schema can be modeled using a UML class diagram, or a number of UML class diagrams grouped by UML packages. Relations between classes are modeled using UML constraints like the cardinality of objects.

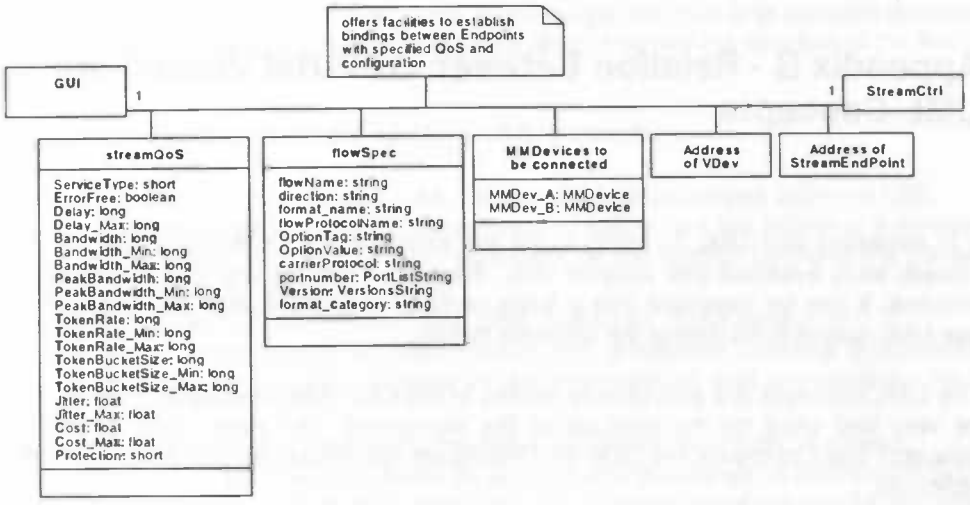


Figure 9-10: ODP-RM Invariant Schema modeled using UML Class Diagram

Because the ODP-RM information viewpoint focuses on the *information* which is passed between the objects, a typical UML class diagram for the ODP-RM information viewpoint consists of showing what information objects are passed between the components of a distributed system. Figure 9-10 shows an example of such a class diagram. The information objects are modeled by UML Association Classes (these are the classes connected through dotted lines to the association between classes modeling components of a distributed system).

An UML note can be used to describe the *contract* between the actors which exchange the information objects

Static Schema

A *static schema* is represented as a UML package of references to one or more UML objects, their UML attribute links, their UML link ends which have an associated target link end which is navigable, their UML classifiers, and, for those objects with UML state machines, a UML dependency to the UML state of the object, with a UML tagged value or a named property list specifying an interval of time.

A static schema is represented as a 'snapshot' of the UML class diagram at a particular point in time, and, for objects with state machines, the current state of the UML state machine associated with that object. A static schema represents one particular state of a dynamic schema

Dynamic Schema

A *dynamic schema* is represented as a UML state machine or a UML package of the state machines of several UML objects

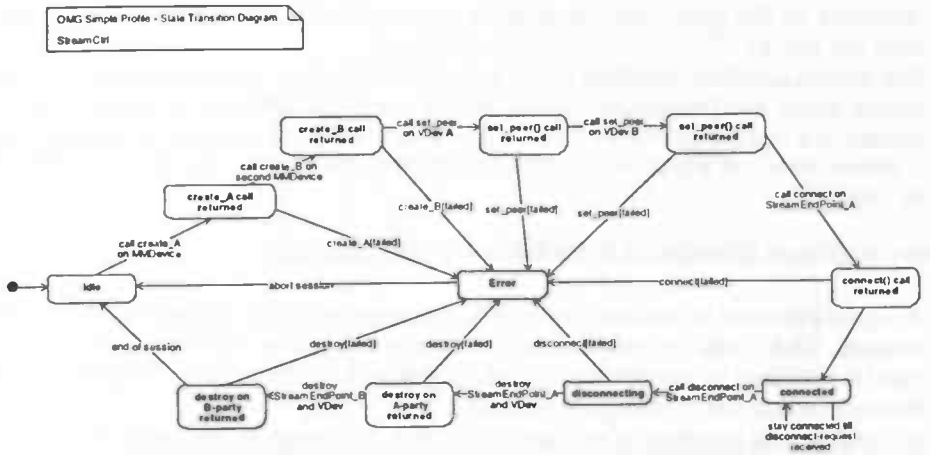


Figure 9-11: ODP-RM Dynamic schema modeled using UML State Machine

A dynamic schema represents the possible state changes of an ODP-RM component, with emphasis on the information which is received by, send from and held by that component. A dynamic schema also shows how information involves and is passed between the different components of the system during the operation of the system. Figure 9-11 shows an example of a UML state machine.

Relations between Computational Viewpoint concepts and UML concepts

A specification from the Computational Viewpoint focuses on the structure of a distributed application and the interaction between the computational objects of which the application consists. The computational specification addresses the following statements [Leydekkers, 1997]:

- What are the computational objects of the system and how are they structured?
- What are the interfaces of the computational objects?
- What operations can be invoked to and from the computational interfaces?
- What is the role of the computational interface?
- What behaviour is observable at the computational interfaces?
- What environment constraints are associated with the computational objects and associated interfaces?
- What interaction is possible between computational objects (interfaces)?

A computational object can be modeled by a UML class. The interfaces of a computational object are modeled by the methods that are defined on a class. Classes and the methods defined on them are grouped in a UML Class Diagram. This Class Diagram is different than the Class Diagram discussed in the previous Section, by focusing on the computational objects and the relations between those computational objects.

The interactions between computational objects takes place via interfaces. The computational specification can be modeled by using three types of operational interfaces:

- The *signal interface*: a signal interface is an elementary interface. It initiates signals and might receive responding signals. A signal is an elementary atomic interaction. A signal interface signature describes the characteristics of each signal.
- The *operational interface*: interface whose signature describes a set of operations. An operation is similar to a procedure and is invoked on designated interfaces. Operations reflect the client/server paradigm. An operation is an interaction between a client object and a server object to execute a function by the server and return the results to the client.
  There are two types of operation: an *interrogation*, in which the server returns a

response on the client request, and an *announcement*, in which there is no response from the server.

- The *stream interface*: interface which describes behaviour, which consists of a single non-atomic action that persists throughout the lifetime of the interface. A stream interface may consist of a number of related flows, either in one single direction or in opposite directions. A stream interface signature describes the type of the flow and the direction of the flow (i.e. its causality).

These interfaces correspond to the following UML concepts:

- A *signal interface* is modeled as a UML message which is passed between UML classes. UML does not incorporate the concept of failure, but the failure of a message can be modeled by specifying an UML constraint on the relation between the classes between which the message is passed.
- An *operational interface* is modeled as a UML message, in the case of an announcement. In the case of an interrogation, the operational interface is modelled as a pair of UML messages: one message for the interrogation, and one message for the answer.
- A *stream interface* cannot be directly modeled using UML concepts. A stream interface, however, can be considered as a set of flow interfaces. A flow interface can again be modeled by a set of UML messages, so a stream interface can be modeled by a set of one or more sets of UML messages.

These concepts can be visualised using the following UML diagrams:

- *Class Diagram* - focuses on the computational objects and the relations between those computational objects. Figure 9-12 shows an example of a Class Diagram (operations are not shown for simplicity reasons).

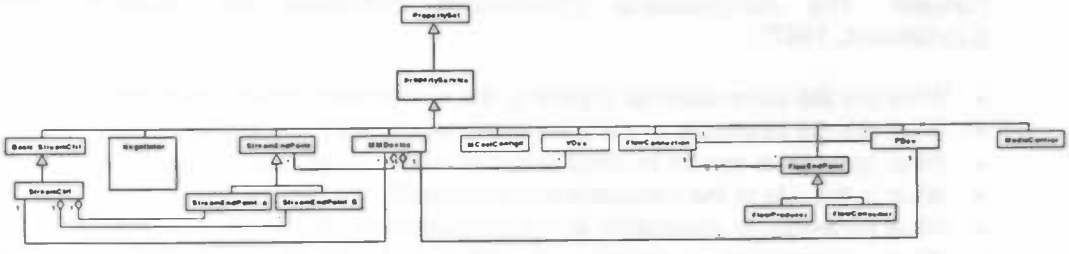


Figure 9-12: UML Class Diagram

- *Collaboration diagram* - focuses on the operation sequence and the structure of the computational objects in a system. Figure 7-4 shows an example of a Collaboration Diagram. This diagram also shows the messages exchanged between the computational objects. The operation sequence is shown by the numbers associated with each message.

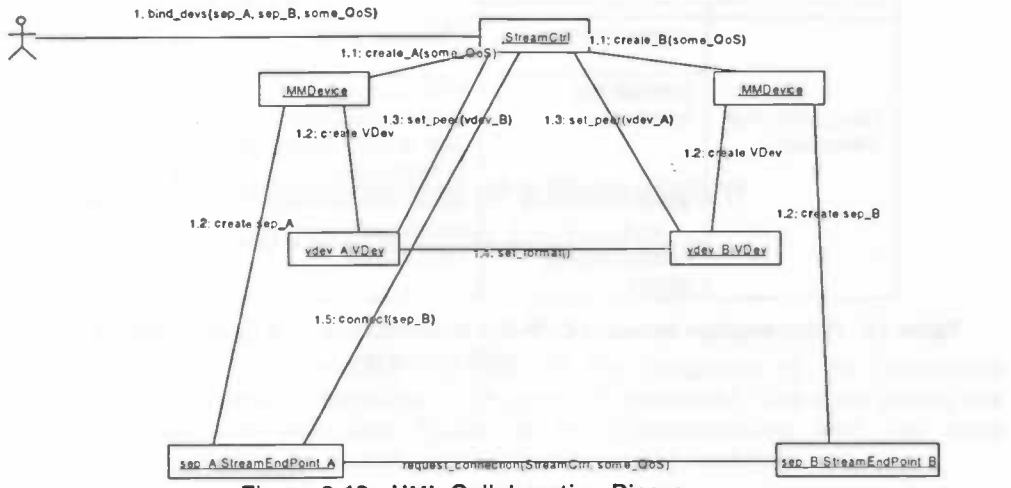


Figure 9-13: UML Collaboration Diagram

- *Sequence diagram* - focuses on the behaviour of the system evolving in time. Figure 7-5 shows an example of a Sequence Diagram.

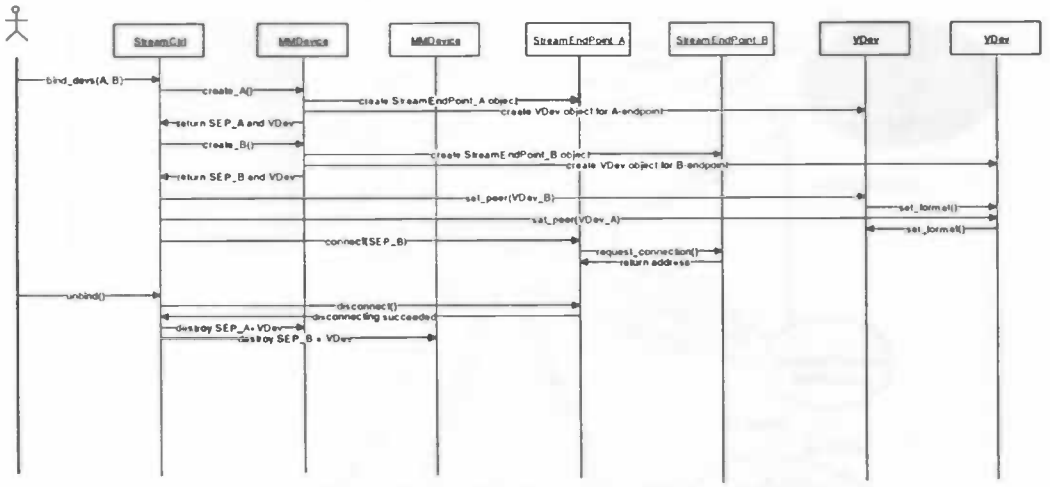


Figure 9-14: UML Sequence Diagram

Table 7 summarises the relationships between the ODP-RM Information and Computational Viewpoints, ODP-RM concepts of these viewpoints, UML concepts, and UML diagrams.

ODP-RM Viewpoints	ODP-RM concepts	UML concepts	UML diagrams
Information Viewpoint	invariant schema	package of references to class objects	Class diagram
	static schema	'snapshot' of the class diagram at a particular point in time, and, for objects with state machines, the current state of the state machine.	Class diagram and (possible) State diagram
	dynamic schema	UML state machine or a UML package of the state machines of several UML objects	State diagram
	information object	UML Association Class	Class diagram

Computational Viewpoint	signal interface	UML message	Collaboration diagram
	operational interface	UML message (announcement) pair of UML messages (interrogation)	Activity diagram Sequence diagram
	stream interface	set of sets of UML messages	Class diagram
	computational object	UML Class	Class diagram

Table 15: Relationships between ODP-RM Informational and Computational Viewpoints and UML concepts

Appendix C - Implementation of a Distributed Videoconferencing Application Endpoint

In this Appendix the implementation of one of the endpoints of the distributed videoconferencing application discussed in Chapter 7 is described. The main points are the operation sequence, important topics in the implementation and the used environments, programming languages and videoconferencing hardware.

The following Figure gives a detailed overview of the components and operation sequence of the implementation:

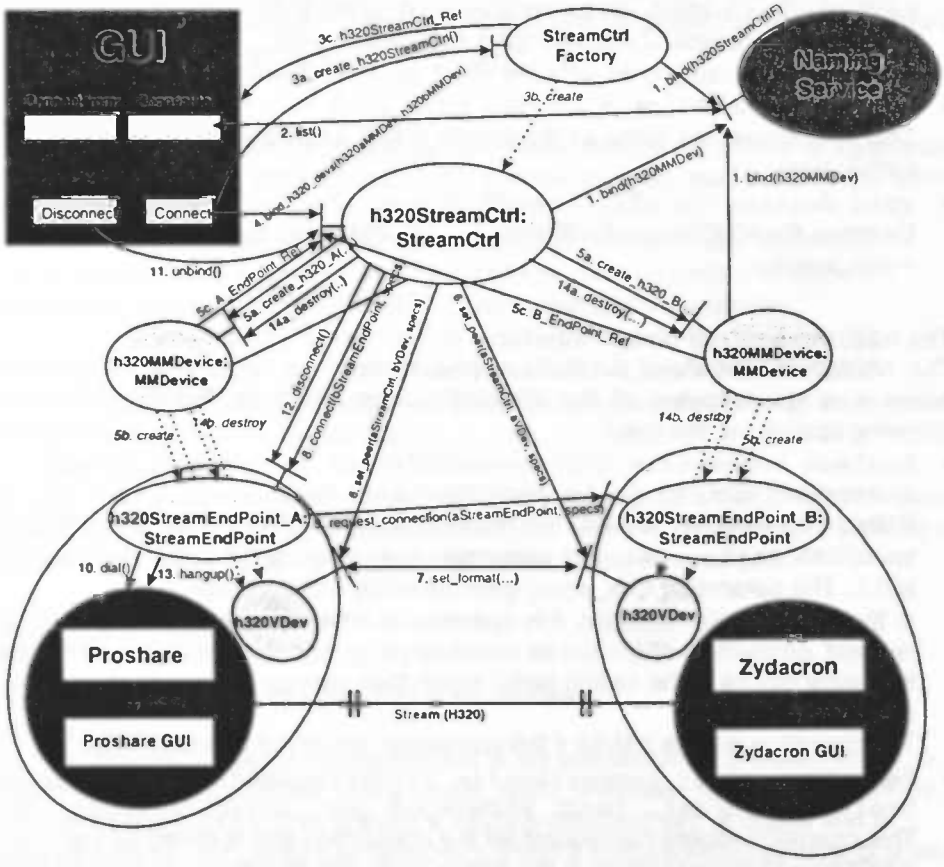


Figure 9-15: Detailed Architecture and Operation Sequence

In this Figure the components described in Chapter 7 can be found. The Figure also shows the operation sequence to set up and break down a binding. The difference is that specialised implementations are made for all components, specially tailored for the H.320 implementation. This way of specialisation is not recommended for extensible implementations, but for this demonstrator this approach is sufficient.

Interfaces

The remainder of this Appendix focuses on the implementation of the 'Proshare-endpoint'. This endpoint consists of the following objects: h320MMDevice, h320StreamEndPoint_A, h320VDev and an object abstracting from a hardware device. These objects have the following interfaces (specified in CORBA IDL):

The h320MMDevice interface

The h320MMDev object is a specialisation of the MMDevice interface. It has the same functionality as the MMDevice interface, the only difference is that the operations *create_h320_A()* and *create_h320_B()* are added.
For the ED4-3 demo, the following operations are implemented:

- `h320StreamEndPoint_A create_h320_A(in h320StreamCtrl the_requester, out h320VDev the_vdev, inout streamQoS the_qos, out boolean met_qos, inout string named_vdev, in flowSpec the_spec)`
  Creates a h320StreamEndPoint_A object and a h320VDev object `the_vdev`. The parameter `the_qos` is not used and gets the value `NULL`. The parameter `named_vdev` will be filled when the call returns and contains the name of the hardware device which will be encapsulated by the VDev, for example "Proshare" for a proshare-device and "Zydacron" for a zydacron-device.
  The parameter `the_spec` gets the value `nilFlowSpec`.
- `h320StreamEndPoint_B create_h320_b(...)`
  This operation is the same as the *create_h320_A* operation, so it will not be discussed further here
- `void destroy(in h320StreamEndPoint the_ep, in string vdev_name)`
  Destroys the h320StreamEndPoint `the_ep` and the associated h320VDev named `vdev_name`

The h320StreamEndPoint_A interface

The h320StreamEndPoint_A interface inherits from the h320StreamEndPoint interface, which is a specialisation of the StreamEndPoint interface. For the ED4-3 demo, the following operations are used:

- `boolean connect(in h320StreamEndPoint responder, inout streamQoS qos_spec, in flowSpec the_spec)`
  Establishes a connection with the h320StreamEndPoint `responder` and flow specifications `the_spec`. The parameter `qos_spec` is not used, and gets the value `NULL`. The parameter `the_spec` gets the value `nilFlowSpec`.
  In the ED4-3 implementation, this operation is implemented by calling the *request_connection*-operation on the other party, and then issuing a *dial*-request to the hardware device of the calling party, which then sets up the actual connection (i.e. it calls the other party)
  The operation returns `FALSE` if the connection could not be established.
- `boolean request_connection(in h320StreamEndPoint initiator, in boolean is_mcast, inout streamQoS qos, inout flowSpec the_spec)`
  This operation makes the request for the connection and is called by the h320StreamEndPoint which is the calling party (the parameter `initiator`) on the h320StreamEndPoint which is the called party. The parameters `is_mcast` and qos are not used and respectively get the values `FALSE` and `NULL`. At the call this parameter holds the flow specifications of the calling party, when the call returns, it holds the flow specifications of the called party.
  In the ED4-3 implementation, the parameter the_spec has the value nilFlowSpec at the call. When the call returns, this parameter has to be filled with the phone number of the called party.
  The operation returns `FALSE` if the request is not granted.
- `void disconnect(in flowSpec the_spec)`
  This operation disconnects the stream connection with specifications `the_spec`. In the implementation this operation issues a *hangup*-request to the hardware device of

the calling party, which then ends the connection.
The parameter `the_spec` gets the value `nilFlowSpec`.

The h320VDev interface
The h320VDev interface is a specialisation of the VDev interface. The functionality of these interfaces is the same, but the h320VDev interfaces uses different parameters in the operations to support the H.320 connection.
For the ED4-3 demo, the following operations are used:

- `boolean set_peer(in h320StreamCtrl the_ctrl, in VDev the_peer_dev, inout streamQoS the_qos, in flowSpec the_spec)`
  This operation is called on both h320VDevs initiates the negotiation procedure between the h320VDevs. The operation has to be first called on the h320VDev of the 'A-party', and then on the h320VDev of the 'B-party'. In the implementation this operation calls the operation *set_format* on the peer h320VDev `the_peer_dev` to retrieve information about their capabilities (these operations are described below). `the_ctrl` is the calling h320StreamCtrl, the other parameters are already discussed in previous sections.
- `void set_format(in string flowName, in string format_name)`
  This operations is called by the operation *set_peer*. The parameter `flowName` is empty because this implementation only deals with one stream. The parameter `format_name` holds the stream format. In this implementation this parameter has the value `"UNS"`, which is an acronym for unspecified.

The hardware devices
Although the applications which control the hardware devices (Proshare or Zydacron) do not have interfaces which are exposed to the outside, they do need to have the following two operations:

- `boolean dial(in string connectionType, in string address)`
  This operation makes a connection to the address `address` with the standard specified by `connectionType`. In this implementation the parameter `connectionType` has the value `"H320"`
  The operation returns `FALSE` if the connection could not be made. The error notification and error handling (e.g. wrong number, line is busy) have to be provided by the implementation of the *dial*-operation
- `boolean hangup()`
  This operation ends the stream connection. It returns `FALSE` if the connection cannot be correctly ended. As with the *dial*-operation, this operation also is reponsible for error notification and error handling when the connection cannot be ended.

Operation Sequence

The operation sequence for the endpoint is the following:
Connection set up:

1. (5a) The StreamCtrl does a *create*-request to the two MMDevs: a *create_h320_A*-request to the calling MMDev, and a *create_h320_B*-request to the called MMDev.
   (5b) The calling h320MMDev creates a h320StreamEndPoint_A object and a h320VDev object, the called h320MMDev creates a h320StreamEndPoint_B object and a h320VDev object.
   (5c) The references to these objects are returned to the StreamCtrl (These references are retumed by the *create_h320_A[B]*-operations).
   Call: `create_h320_A[B](the_h320StreamCtrl, the_vdev, NULL, met_qos, "Proshare"["Zydacron"], nilFlowSpec)`
2. The h320StreamCtrl does a *set_peer*-request on the VDev objects to initiate the negotiation procedure between the VDevs to set up a connection with the requested specifications.
   Call: `set_peer(the_h320StreamCtrl, the_b[a]VDev, NULL, nilFlowSpec)`
3. The VDevs configure the requested connection parameters by calling the operation *set_format* and on the peer VDev. These operations check if all the flows originating from a VDev can be understood by the corresponding flow consumers on the peer

VDev.
Additional configurations can be accomplished by calling the operation *configure*.
Call: `set_format(flowName, "UNS")`

4. The StreamCtrl issues a *connect*-request to the StreamEndPoint_A stream endpoint to establish a connection with desired specifications.
   Call: `connect(the_h320StreamEndPoint_B, NULL, nilFlowSpec)`
5. The StreamEndPoint_A object sets up the connection by doing a *request_connection*-request on the StreamEndPoint_B.
   Call: `request_connection(the_StreamEndPointA, FALSE, NULL, nilFlowSpec)`
   Note that the phone number of the B party has to be returned in the last parameter
6. The StreamEndPoint_A object establishes a stream connection between the stream endpoints.
   Call: `dial("H320", address)`. The parameter `address` is a string (see also section 0).

Disconnecting:

12. The h320StreamCtrl then does a *disconnect*-request to StreamEndPoint_A
    Call: `disconnect(nilFlowSpec)`
13. The StreamEndPoint_A objects issues a *hangup*-request to the associated hardware device
    Call: `hangup()`
14. The h320StreamCtrl calls the *destroy*-operation on the h320MMDevs, which destroy the h320StreamEndPoint and h320VDev objects
    Call: `destroy(the_h320StreamEndPoint_A[B], the_VDev)`

Used Platforms, Programming Languages and Hardware

The following platforms, programming languages, CORBA implementations and video hardware devices are used for the implementation. Note that there are three end user domain descriptions, each describing a different configuration used.

GUI

- Platform: Internet Browser supporting Java v.1.1
- Programming Language: Java v.1.1
- CORBA implementation: IONA OrbixWeb v.3.0

Provider (StreamCtrlFactory and StreamCtrl components)

- Platform: Unix (SUN Solaris)
- Programming Language: Java v.1.1
- CORBA implementation: IONA OrbixWeb v.3.0

End User Domain 1 (h320MMDevice, h320StreamEndPoint_A, h320VDev and hardware interface components)

- Platform: Windows 95
- Programming Language: Microsoft Visual C++ 5.0, Intel Proshare Developer's Kit v.3.0
- CORBA implementation: IONA Orbix v.2.3c
- Video Hardware: Intel Proshare

End User Domain 2 (h320MMDevice, h320StreamEndPoint_A, h320VDev and hardware interface components)

- Platform: Windows NT
- Programming Language: Microsoft Visual C++ 5.0
- CORBA implementation: IONA Orbix v.2.3c
- Video Hardware: Zydacron

End User Domain 3 (h320MMDevice, h320StreamEndPoint_A, h320VDev and hardware interface components)

- Platform: Windows NT

- Programming Language: Microsoft Visual C++ 5.0, Intel Proshare Developer's Kit v.3.0
- CORBA implementation: Chorus CoolOrb v.4.1
- Video Hardware: Intel Proshare

Image of the Endpoint User Interface