Distributed Multimedia in Telecommunications Service Engineering Using the Distributed Component Object Model

DIONISIS X. ADAMOPOULOS
Centre for Communication Systems Research
School of Elec. Eng., IT and Mathematics
University of Surrey
Guildford GU2 5XH
ENGLAND

GEORGE PAVLOU
Centre for Communication Systems Research
School of Elec. Eng., IT and Mathematics
University of Surrey, England
Guildford GU2 5XH
ENGLAND

CONSTANTINE A. PAPANDREOU
Hellenic Telecommunications Organisation (OTE)
17 Kalliga Street, GR-114 73 Athens
GREECE

Abstract :- Nowadays, the demand for a great variety of sophisticated telecommunications services with multimedia characteristics is on the increase. This trend highlights the need for the efficient creation of distributed programs with multimedia data exchange running on distributed processing environments. Therefore, it is necessary to support the object-oriented development of distributed multimedia applications in a flexible manner. This paper recognises Microsoft's DCOM as a key technology solution in the area of service engineering and examines an RM-ODP compliant approach to enhance it for the handling of continuous media flows through the design and implementation of a collection of suitable multimedia support services. These services are validated through a number of simple scenarios, and their alignment with TINA-C, together with the examination of QoS issues are also coniere d.IMACS/IEEE CSCC'99 Proceedings, Pages:5051-5055

Key - Words :- Distributed multimedia, DCOM, new telecommunications services, distributed object computing.

1 Introduction

Driven by technological advancements, market growth and deregulation, the global telecommunications environment is rapidly adopting a highly dynamic and open character, which, in combination with the evolving synergy between information and telecommunication technologies, provides a wide range of opportunities for the delivery of advanced multimedia telecommunications services (telematic services). Due to recent developments in broadband networks, object orientation, and distributed computing, these telecommunications services are designed and created as distributed multimedia applications in distributed object platforms in a steadily growing rate [1] [9].

Despite the fact that multimedia support has been considered in general terms in the ISO's / ITU-T's Reference Model for Open Distributed Processing (RM-ODP) [5], it is not specified in Microsoft's Distributed Component Object Model (DCOM) [2], and is not yet mature in the Object Management Group's Common Object Request Broker Architecture (CORBA) [6] [10]. On the other hand, a wide range of new telecommunications services are becoming increasingly popular by employing audio and video to convey information or to enhance communication among human users (e.g. videoconference, video on-demand, etc.). Therefore, in the emerging telecommunications environment it is necessary to facilitate the rapid and flexible deplo yment of a great diversity of multimedia, multi-party services by providing direct support to continuous media in Distributed Processing Environments (DPEs).

This paper considers an approach that extends DCOM into an environment suitable for the development of advanced multimedia telecommunications services. More specifically, it examines central issues associated with the provision of object-oriented support for the handling (representation, transmission, and management) of continuous media in DCOM.

2 Modelling Multimedia Telecommunications Services

Multimedia communication involves the interaction of devices which can deal with networked suppliers and consumers of various t ypes of digitally repre-

sented information. Broadly, the tasks involved in this process can be divided into the proper coding and transport of the different media, and into related control aspects, such as how to locate services, request transfer, establish and maintain connections, ensure integrity and timeliness, and control presentation aspects of the delivery of multimedia information. These control aspects, together with the structure and the functionality of the related control software, are the concern of this paper, as they are particularly important for the realisation of the full potential of distributed object platforms [9].

Streams play a central role in the control software. The model of object interaction conventionally adopted in distributed object platforms (i.e. operation invocation) is inappropriate for continuous or dynamic media. Thus, for these media types, a streaming mode of communication is required (continuous interactions) rather than a request / reply-based operation invocation model [3] [7]. This is also in full agreement with the RM-ODP's multimedia computational model [5].

The control software of new telecommunications services which exploit continuous media, has to [3] [9] [13]:

- Model continuous media communications.
- Comply with standards.
- Allow and support a range of Qualities of Service (QoS).
- Support synchronisation of continuous media.
- Enable communication of continuous media to groups of users.
- Support careful management of resources.

These requirements highlight the need for a flexible approach in the development and deployment of open standards-based media exchange telecommunications services. Thus, the proposed approach recognises flexibility as the key requirement and concentrates on the problem of providing a flexible way of handling continuous media in DCOM. This problem is addressed by providing a generic platform (multimedia support platform) of primitive COM objects or services (multimedia support services), which can facilitate the construction of new telecommunications services with multimedia characteristics, and by providing an efficient way for managing the multimedia support services within the multimedia support platform.

More specifically, the multimedia support platform provides mechanisms for creating and manipulating COM objects using a number of support services. These services, and the related objects, are compatible with RM-ODP, enabling thus a wide degree of information sharing and application interoperability. The interfaces provided by these services have been designed to allow the underlying technologies maximum flexibility in achieving their implementation.

3 The Distributed Component Object Model

DCOM is the distributed extension to COM (Component Object Model) that builds an Object Remote Procedure Call (ORPC) layer on top of DCE RPC to support remote objects. Thus, DCOM provides an efficient and effective solution for the integration of heterogeneous components in a distributed environment.

However, DCOM does not satisfy directly the more complicated and stringent requirements of multimedia telecommunications services. To enable DCOM to be the basis for new telecommunications services that require the handling and control of continuous media, some extra features are necessary. The most obvious requirement is that the concept of streams be added to the DCOM object model. At present only operational interfaces are defined, which impose a Remote Procedure Call (RPC) model of interaction.

Before focusing on DCOM, it has to noted that COM handles multimedia information through the Microsoft DirectShow architecture (previously Microsoft ActiveMovie architecture), which incorporates the notion of streams. Obviously, the use of this notion is restricted to the environment of stand-alone multimedia capable computers with Microsoft Windows operating systems (9x, NT 4.x, 2000), as DirectShow is not a distributed architecture.

4 Supporting Continuous Media in DCOM

The development of new telecommunications services with multimedia characteristics in DCOM is proposed to be facilitated by a generic multimedia support platform with the introduction of a number of new multimedia support services (DPE services) into the DCOM architecture. These services (which are used in conjunction with existing DCOM services) provide new functionality without requiring any changes to the basic architectural model of DCOM. The new services consist principally of two types of COM objects: devices and stream binders. These are both seen by the higher layers as normal services with standard abstract data type interfaces, but they encapsulate the control and transmission of continuous media.

Devices are an abstraction of physical devices, stored continuous media or software processes. They may be either sources (producers), sinks (consumers)

or transformers of continuous media data. Most devices present a device dependent interface, a generic control or chain interface (IChain), and an endpoint interface (IEndpoint). The device dependent interface contains operations specific to the device modelled (e.g., a camera might have operations such as focus, pan or tilt). Furthermore, it has to be noted that all devices (and every COM object) have to inherit from the IUnknown interface, which provides functionality required by all COM objects.

```
interface IChain : IUnknown
{ typedef enum {in, out, inout}
          DeviceTypes;
  HRESULT GetDeviceType(
          [out] DeviceType* DType);
  HRESULT Start();
  HRESULT StartEx(
          [in] int NumberOfSegments);
  HRESULT Stop();
  HRESULT Suspend();
  HRESULT SuspendEx([in] int Time);
  HRESULT SuspendEx1(
          [in] int NumberOfSegments);
  HRESULT Resume();
  HRESULT Skip([in] int NumberOfSegments);
  HRESULT GetPosition(
          [out] int* SegmentNumber);};
```

Fig. 1: The IChain Interface.

A piece of continuous media can be visualised as a chain comprising a sequence of segments or links, each of which represents an atomic unit particular to the media type in question (e.g. a frame of video) [3]. Thus, a chain is an abstraction over a continuous media source or sink that focuses on the control of the production and consumption of continuous media data. Based on this abstraction, the IChain interface provides generic operations for controlling continuous media devices and managing continuous media transmissions. It is a device independent interface which is common to all continuous media devices. The IChain interface is summarised in Fig. 1 using (a simplified variation of) Microsoft's Interface Definition Language (MIDL), which is an extension of DCE's IDL.

```
interface IEndpoint : IUnknown
{ HRESULT GetSegment([out] BSTR* Segment);
  HRESULT PutSegment([in] BSTR* Segment);
  HRESULT SetCharacteristics(
    [in] long ChrSize,
    [in, size_is(ChrSize)] long* ChrArray);
  HRESULT GetCharacteristics(
    [in, out] long* ChrSize,
    [out,size_is(ChrSize)] long* ChrArray);
};
```

Fig. 2: The IEndpoint interface.

Another interface which is common to all continuous media devices (device independent interface) is the IEndpoint interface. An endpoint is a connection point for a stream, and the IEndpoint interface is actually a stream interface. The IEndpoint interface abstracts over all aspects of a device which are concerned with the transport of continuous media. Essentially, as it can be seen in Fig. 2, it presents a pair of operations, GetSegment and PutSegment through which segments can be read or written respectively.

In order to be able to control streams the binding process must be made explicit [3]. This is done through the introduction of a binding COM object (StreamBinder). StreamBinder represents the connection between bound objects and provides an operational interface (IStreamBinder) through which the binding between streams can be created, monitored, and controlled.

```
interface IStreamBinder : IUnknown
{ HRESULT StartSource(
          [in] IUnknown* SourceGroup);
  HRESULT StartSink(
          [in] IUnknown* SinkGroup);
  HRESULT Connect&Transfer(
          [in] IUnknown* SourceGroup,
          [in] IUnknown* SinkGroup);
  HRESULT StopSource(
          [in] IUnknown* SourceGroup);
  HRESULT StopSink(
          [in] IUnknown* SinkGroup);
  HRESULT SuspendSource(
          [in] IUnknown* SourceGroup);
  HRESULT SuspendSink(
          [in] IUnknown* SinkGroup);
  HRESULT ResumeSource(
          [in] IUnknown* SourceGroup);
  HRESULT ResumeSink(
          [in] IUnknown* SinkGroup);
  HRESULT DestroyConnection(
          [in] IUnknown* SourceGroup,
          [in] IUnknown* SinkGroup);};
```

Fig. 3: The IStreamBinder interface.

More specifically, as can be seen in Fig. 3, the IStreamBinder interface contains operations which allow the client of a StreamBinder to connect and disconnect devices via their IEndpoint interfaces (and thus create and destroy stream connections), start and stop the flow of continuous media information, and suspend / resume the activity of the involved devices. With these operations the StreamBinder hides continuous media transmissions, which can be optimised by using dedicated transport protocols entirely distinct from those used to convey control messages. The binding action can be initiated

by a COM object involved in the binding or by one separate from it.

In the case where it is desirable to start, stop, establish, and generally perform control operations to a number of streams simultaneously, the notion of object groups simplifies greatly the necessary code (calls to the StreamBinder operations) and ensures that the code reflects the correct / intended semantics (it decreases the possibility of missing, wrong, or out of - logical - order operations on devices).

```
interface IObjectGroup : IUnknown
{ HRESULT Join([in] IUnknown* refiid);
  HRESULT Leave();
  HRESULT Use([out] IUnknown* refiid);
  HRESULT Reset();};
```

Fig. 4: The IObjectGroup interface.

Conceptually, object groups are modelled using the COM class ObjectGroup, which collects in a group a set of related COM objects. Actually, it maintains a list of the interface references (REFIIDs) of the COM objects that belong to a specific group. The IObjectGroup interface can be seen in Fig. 4. In a typical scenario, two instances of the ObjectGroup COM class are used: a SourceGroup (with the REFIIDs of the sources) and a SinkGroup (with the REFIIDs of the sinks).

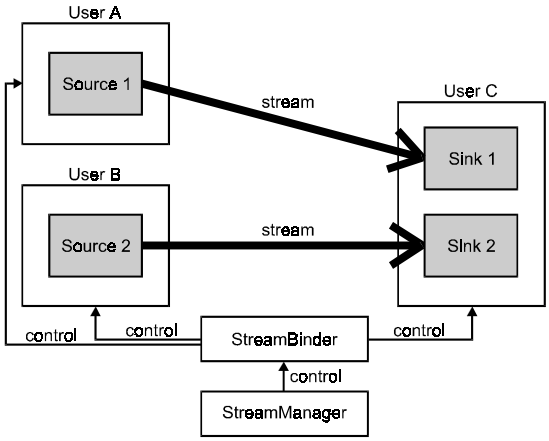

Fig. 5: A scenario for the multimedia support services.

Fig. 5 depicts the COM objects involved in the configuration of a possible scenario for the proposed multimedia support services. In this scenario, two source devices (e.g. videocameras) have been connected via a StreamBinder to two sink devices (e.g. VDUs). Two different streams have been established between the source and sink devices.

The necessary steps that have to be followed in order to realise the two video connections between the sources and sinks of Fig. 5 using the proposed multimedia support services are the following:

Step 1 (Obtain the necessary interface references): The REFIIDs of the two sources (Source1UserA and Source2UserB) and the two sinks (Sink1UserC and Sink2UserC) involved in stream communication are obtained. Device dependent operations are also performed if necessary.

Step 2 (Create new instances of required services): A StreamBinder instance is created and the related interface reference is obtained. Additionally, two ObjectGroup instances are created and the related interface references are also obtained (SourceGroup and SinkGroup).

Step 3 (Form the appropriate object groups (if required)): Taking into account the streams that is desirable to be established, the REFIIDs of the sources become members of the SourceGroup [Join(Source1UserA), Join(Source2UserB)], and the REFIIDs of the sinks become members of the SinkGroup[Join(Sink1UserC),Join(Sink2UserC)].

Step 4 (*Start the devices*): The sink and source devices are started [StartSink(SinkGroup), StartSource(SourceGroup)].

Step 5 (Establish connections between source and sink devices): Associate the appropriate sources and sinks and initiate continuous media transfer between them [Connect&Transfer(SourceGroup,SinkGroup)]. Steps 4 and 5 can also take place in the opposite order.

Step 6 (Stop the devices): When the interaction is finished the sink and source devices are stopped [StopSink(SinkGroup),StopSource(SourceGroup)].

Step 7 (Destroy connections and services): The connections established between the source and sink devices are destroyed [DestroyConnection(Source Group,SinkGroup)]. Then, the StreamBinder and the source and sink groups are also destroyed.

The above described steps constitute a kind of algorithm for establishing stream connections (stream communication algorithm). As it is described in Fig. 5, the StreamManager COM object interacts with the StreamBinder and performs all the steps of the stream communication algorithm. The interface and the functionality of the StreamManager depends on the requirements of the specific application. On the contrary, the interfaces and the functionality of the devices and the StreamBinder are application independent and thus reusable. Actually, these interfaces can be considered as a high level Application Programming Interface (API) for the handling of continuous media in DCOM.

6 Validation and Future Extensions

The proposed multimedia support services and the related API have been tested in several scenarios

involving continuous media transmission with different configurations of COM objects. It has been found that they constitute a viable, flexible, consistent, coherent, and smart way of building multimedia telecommunications services in DCOM.

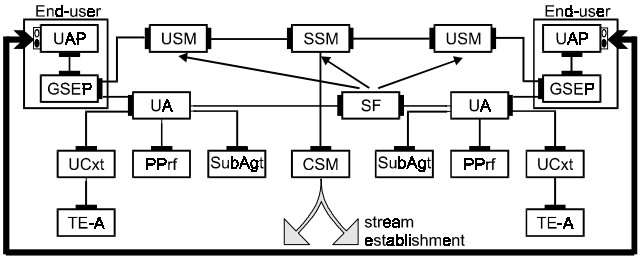

Fig. 6: The computational view of the MMCS for education and training in TINA-C.

In order to validate this conclusion under more realistic conditions of use, the proposed multimedia support services are currently being applied to the development of a MultiMedia Conferencing Service (MMCS) for education and training [11], according to the architectural framework proposed by the Telecommunications Information Networking Architecture Consortium (TINA-C) [8]. The main TINA-C computational objects involved in the MMCS can be seen in Fig. 6. From this figure is also evident that the Communication Session Manager (CSM) incorporates the functionality of both the `StreamManager` and the `StreamBinder`. Thus, in the process of realising this component the conformance of the proposed approach with the TINA-C Stream Channel Model is being examined [12].

Furthermore, the proposed multimedia support services and the related API will be extended to cover Quality of Service (QoS) issues, as it is essential that the service developer is able to select and control the QoS with which connections are made [3]. For this reason, the `StreamBinder` will be enhanced with the ability to specify a number of QoS parameters. However, although the `StreamBinder` specifies these parameters, it is important to note that whether this QoS can be obtained ultimately depends upon whether the underlying transport protocol and network can support the chosen parameters [7].

7 Conclusions

There is a technology push in the area of multimedia communications, which is acting as a catalyst for the specification of new multimedia telecommunications services. Due to efficiency reasons, these services are destined to be deployed in a distributed object environment. Thus, there is an increasingly important need for distributed object platforms to support continuous media interactions in a flexible manner.

In this paper, a number of RM-ODP compliant multimedia support services together with a related API are proposed to enhance DCOM with continuous media support. The viability of this approach is evaluated using a number of simple scenarios. Furthermore, the development of a MMCS for education and training according to TINA-C will ensure that with the proposed approach distributed multimedia objects in DCOM can be controlled and interoperated over a network in a relatively simple and straight-forward way. This alignment with TINA-C, together with the introduction of QoS considerations is expected to enable the proposed support services to open the way for the emergence of broadband multimedia telecommunications services in DCOM.

References:

[1] Adamopoulos, D.X., Papandreou, C.A., Distributed Processing Support for New Telecommunications Services, *Proceedings of ICT '98*, Vol. III, June 1998, pp. 306-310.

[2] Brown, N., Kindel, C., *Distributed Component Object Model Protocol - DCOM/1.0*, Microsoft Corporation, November 1996.

[3] Coulson, G., Blair, G.S., Davies, N., Williams, N., Extensions to ANSA for Multimedia Computing, *Computer Networks and ISDN Systems*, Vol. 25, 1992, pp. 305-323.

[4] Gay, V., Leydekkers, P., Multimedia in the ODP-RM Standard, *IEEE Multimedia*, Vol. 4, No. 1, 1997, pp. 68-73.

[5] IONA Technologies, *Orbix MX: A Distributed Object Framework for Telecommunication Service Development & Deployment*, April 1998.

[6] Kinane, B., Muldowney, D., Distributed Broadband Multimedia Systems Using CORBA, *Computer Communications*, Vol.19, 1996, pp. 13-21.

[7] Magedanz, T., TINA - Architectural Basis for Future Telecommunications Services, *Computer Communications*, Vol. 20, 1997, pp. 233-245.

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

[9] Object Management Group, *Control and Management of Audio / Video Streams*, OMG document telecom/97-05-07, October 1997.

[10] Papandreou, C.A., Adamopoulos, D.X., Design of an Interactive Teletraining System, *British Telecommunications Engineering*, Vol. 17, Part 2, August 1998, pp. 175-181.

[11] TINA-C, *TINA-C Stream Channel Model*, Document No.TR_MFJ.001_1.4_95, Feb. 1996.

[12] Waddington, D., Coulson, G., A Distributed Multimedia Component Architecture, *Proceedings of EDOC '97*, 1997, pp. 17-35.