MMFramework - Distributed Environment for Control and Management of Multimedia Data

Krzysztof Zieliński, Łukasz Czekierda, Piotr Nawrocki, and Mariusz Michalski

Dept. of Computer Science, University of Mining & Metallurgy
Al. Mickiewicza 30, Cracow, Poland
tel:+48 (12) 617 39 82, fax:+48 (12) 617 39 66
{kz, luke, piter, cmich}@cs.agh.edu.pl
http://www.cs.agh.edu.pl

Abstract. This paper presents an open multimedia framework, called MMFramework. It may be used for dynamic creation of various multimedia applications. The proposed framework architecture consists of six layers. Its definition results from decomposition of the system into components with well-defined interfaces and internal implementation dedicated to the given hardware usage. Each layer represents a collection of components which are characterized by similar functionality. The system has been designed with support of existing CORBA Services and OMG specifications related to multimedia.

1 Introduction

The multimedia applications still represent important area of system research which addresses increasing demand for expanded functionality, higher quality, portability and flexibility. These requirements meet with common acceptance of new paradigms of distributed computer systems design and implementation built around distributed objects and thin clients dynamically loadable on demand. The distributed systems technology break-through appeared at great extent thanks to OMG efforts of CORBA standardization, and the spread of SUN-Soft Java. Both technologies despite theirs definition of general information processing architectures offer some support for multimedia. The OMG specification "Control and Management of Audio/Video Streams" (OMG A/V) [6, 7] defines a model for implementing an open distributed multimedia streaming framework while SUN-Soft supports JMF (Java Media Framework) [9] dedicated for multimedia GUI building. Neither of these proposals provides complete system support for implementation of distributed multimedia applications.

This paper presents an open multimedia framework, called MMFramework, which may be used for dynamic creation of various multimedia applications and which could be extended by new multimedia devices. The proposed framework architecture consists of six layers. Its definition results from decomposition of the system into components with well-defined interfaces and internal implementation

dedicated to the given hardware usage or applied policy of the system control and management. Each layer represents a collection of components which are characterized by similar functionality. The system has been designed with existing CORBA Services and OMG specifications related to multimedia applications in mind.

The structure of the paper is as follows. In Section 2 the multilayer structure of the MMFramework will be presented. Section 3 concentrates on the Management and Access Control Layer functionality as the only layer of the defined framework directly visible by the user. Next in section 4 the communication mechanisms between components belonging to different layers have been analized in detail. Section 5 defines a procedure of attaching new devices to the system. Section 6 elaborates the example of GUI for dynamic construction of multimedia applications based on the functionality of the proposed architecture. The paper ends with conclusions.

2 MMFramework architecture

The proposed decomposition of the MMFramework into layers is depicted in Fig. 1. The detachment of the three bottom layers is straightforward. These layers provide full access to the multimedia devices configuration and usage. The complexity of the second and third layer is covered by layers four and five. These layers construct a uniform view of the system providing an easy to use abstraction of system components seen from the top application layer.

The first layer called **MMHardware and System Software Layer** deals with multimedia hardware and software provided by vendors. This layer is represented by a wide spectrum of devices such as: video cameras, computers, audio/video encoders/compressors (such as our NPoint device), media servers (for example Sun MediaCenter server), etc. These devices are usually equipped with proprietary control software. From the user's point of view, this layer looks like an inconsistent collection of different devices.

The second layer — **MMHardware CORBA Server Layer** packs up the vendor-provided software with CORBA interfaces. This layer introduces a uniform abstraction of the service provided by a given device from the lower layer. This abstraction is defined by an interface specified in IDL and standard communication mechanisms provided by the IIOP protocol. The IDL interfaces defined in this layer support all operations provided by the native software. The main goal of introducing this layer is to establish a common base for the system development. The choice of CORBA distributed objects model was one of the most significant decisions for the MMF architecture.

The third layer — **A/V Streams Control Layer** is dedicated to multimedia streams creation, control, and destruction. This layer implements the OMG specification [6, 7] and provides software objects which expose functionality of the lower layer CORBA servers in a standard form most suitable for audio and video streams control. It provides an abstraction of a stream encapsulated in the form of a CORBA object which represents its parameters and control opera-

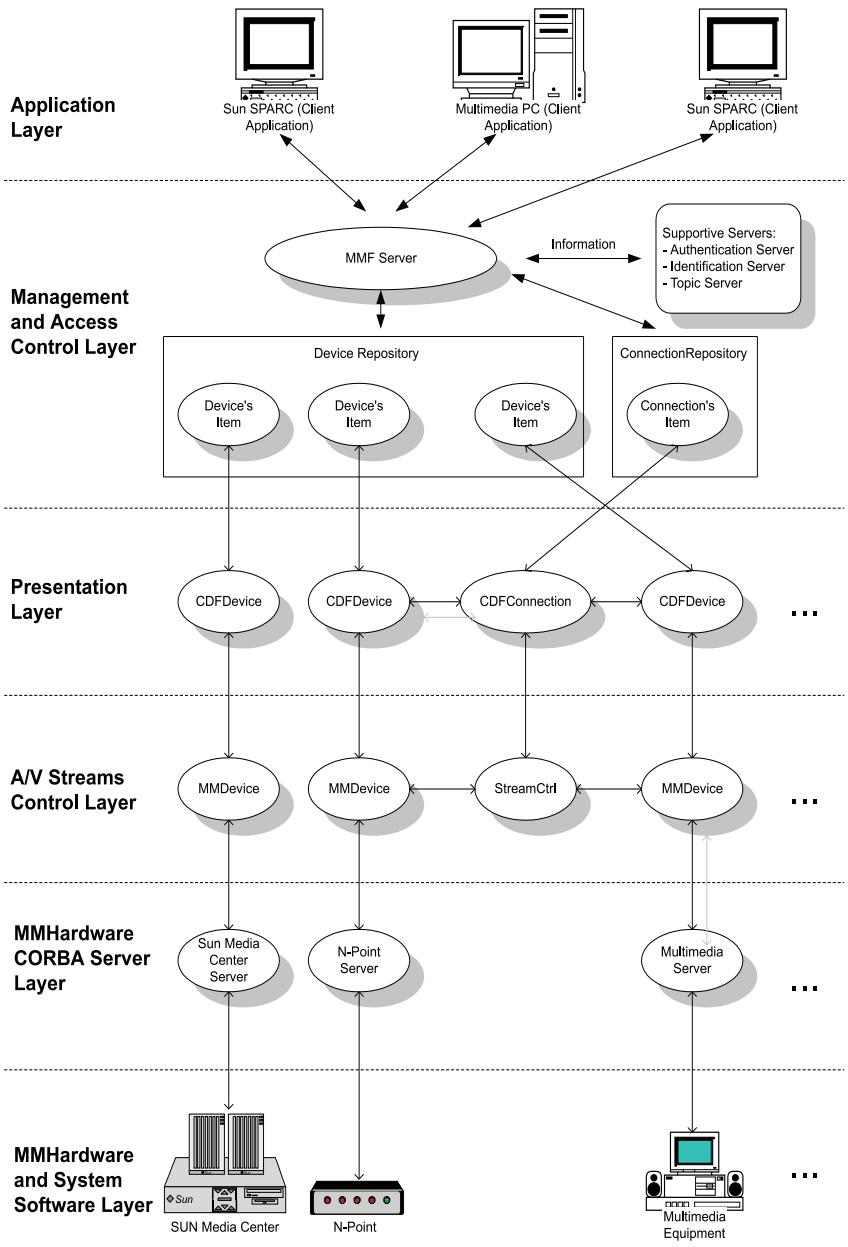

Fig. 1. Multilayer MMFramework architecture

tions. This layer deals not only with the streams control activity but also provides mechanisms for streams parameters negotiation between source and destination multimedia devices and provides streams addressing and QoS control.

The fourth layer — **Presentation Layer** resolves the problem of different types of data used for the parameters of multimedia devices and streams representation. The main goal of this layer is to translate the parameters types from their actual values to especially constructed CDF (Common Data Format). This format is used above the Presentation Layer to simplify presentation of the system's state and to provide a uniform view of the system components for control and visualization purposes. This layer supports users with a powerful mechanism of forms what makes the system configuration simple and less susceptible to errors.

The **Management and Access Control Layer** provides a uniform view of the MMF components' state and a set of functions for manipulating and accessing them. The most important design decision about the construction of this layer is the choice of a model for representation of the system's state components. The most natural representation of the system is a repository of observable objects. They are objects with their own interfaces and notification mechanism. There are two types of items: devices and connections.

A top layer of the MMF architecture is called **Application Layer**. The entities of this layer are collections of user interfaces that provide access to control and visualization of the system state in the most convenient form. The objects defined on this level act as observers of the system components and combine them in the given application scenario. They may also perform the MMF clients' role actively changing the system's state by operations invocations on the devices and connections abstraction provided by the lower layer.

3 Management and Access Control Layer

The Management and Access Control Layer is the only layer visible by the user applications. This influences the tasks of the layer. It has to deliver a compact and uniform interface for end users, that offers access to the whole functionality of the system. The next its task is to provide methods of informing about the actual state of the system (see next sections).

Figure 1 illustrates the key components of the layer. It consists of two types of the repositories:

- Device Repository, which collects objects representing multimedia devices, and
- Connection Repository, containing objects which describe and control the state of multimedia streams connections,

and so called Supportive Servers.

An item from the Device Repository is variously comprehended. This may be (i) 'real' multimedia device, e.g. video server or hardware MPEG coder, (ii) multimedia workstation with its hardware and software enabling sending/receiving

audio and video streams, (iii) *mobile device*, which is ascribed to a system's user and associated at the given moment with one of the workstations. Despite this diversity, the items are treated uniformly from the Management Layer's point of view.

Items from the Connection Repository represent the state of the connection between two or more devices as well as allow performing operations on the streams, like `start` or `stop`.

The main objective during the Management Layer's design was to create one logically centralized object, MMFServer, having knowledge about the whole system. That makes it possible to efficiently organize (i) an access to the system functions, (ii) monitoring its state, and (iii) making this state available to users in case of their synchronous invocations. The client never has direct access to the repository item. The only object the client has access to is MMFServer. It is necessary to point out that notification information is sent to observers by each of repository items separately. The notification information is passed directly to the observers in the Application Layer, hence the MMFServer does not represent the centralized point of the system what could made a bottleneck.

As has been already mentioned, there is a number of so-called supportive servers also considered as a part of the Management and Access Control Layer. They are designed in such a way as to give an external support for items' management and creation and to provide to the Connection and Device Repositories some external services described below.

One of supportive servers is an Identification Server. Items are externally identified by theirs identifiers. The id parameter is translated by the Identification Server to the item's reference which is used for further operation processing.

External support is also needed for the access control function which is performed by an Authentication Server. Operations requiring authentication have an extra parameter — a user's key. The key is used by the Authentication Server to check user's permission to perform a given operation. Thus the repository items don't need to have knowledge about the users and their rights.

As can be seen, it is a role of the MMFServer to contact the Authentication Server and the Identification Server and then invoke an operation on the repository item.

The third supportive server — Topic Server is to handle topics used by OrbixTalk [2] — IONA's implementation of OMG's Event Service. These attributes concern the control of notification event channels' space. A number of such channels can be associated with each item. Using the channels the receivers can be precisely informed about actual state of the system taking advantage of the fact that each topic represents some aspect of the item's state. The used *push* model of information dissemination allows the receiver to implement the response to the notification message in the suitable way.

4 Information flow in MMFramework

Structuralisation of MMFramework in the form of observable objects' repository induces the following model of the information flow:

1. The changes of the device or connection settings are propagated from Application Layer to Management and Access Control Layer. Then the parameters in the CDF format are changed into hardware layer parameters such as QoS (by mechanisms of Presentation Layer).
2. All parameters are further propagated through layers (from the Presentation Layer to the MMHardware CORBA Layer) in form of QoS parameters and Properties (such as transmission parameters). These parameters are finally transformed into operation invocation(s) on CORBA servers from the MMHardware CORBA Server Layer. If the operation is possible to be executed, confirmation or the results are sent in response, if not — the negative answer is sent.
3. The results of the operations are propagated from hardware layer to the Presentation Layer. These results of the changes in the devices' or connections' states are transformed by the presentation server into the CDF form and reported to the device's item or the connection's item respectively.
4. The clients' objects may be registered as the observers of the change of the states of specific items from MaACL layer. All these observers are informed by the Event Channel mechanism about changes of devices' or connections' parameters. All observers are informed about positive results while only the client responsible for the action is informed about negative results.

It is easy to see that the information flow up the multi-layer hierarchy in MMF is provided by two mechanisms: (i) notifications, and (ii) responses to operation calls. The first mechanism corresponds to push monitoring model and the second is pull style of data collection. The flow down the hierarchy is performed only by the operation invocations.

The UML diagram in Fig. 2 shows a sequence of operations' invocations when a new connection is set. The main operations are *initConnection* which prepares connection's parameters and *setConnection* which invokes operations on objects in A/V Streams Control Layer. This UML diagram presents interaction between (from the left hand side) five objects (MMFServer, Authentication Server, Identification Server, Connection, TopisServer) which belong to Management and Access Control Layer and objects (CDFConnection, CDFDevice1, CDFDevice2) belonging to the Presentation Layer and the last three objects (StreamCtrl, MMDevice1, MMDevice2) which are contained in A/V Streams Control Layer. This illustrates the information flow (in the form of operations' invocations) between the layers of the MMFramework system.

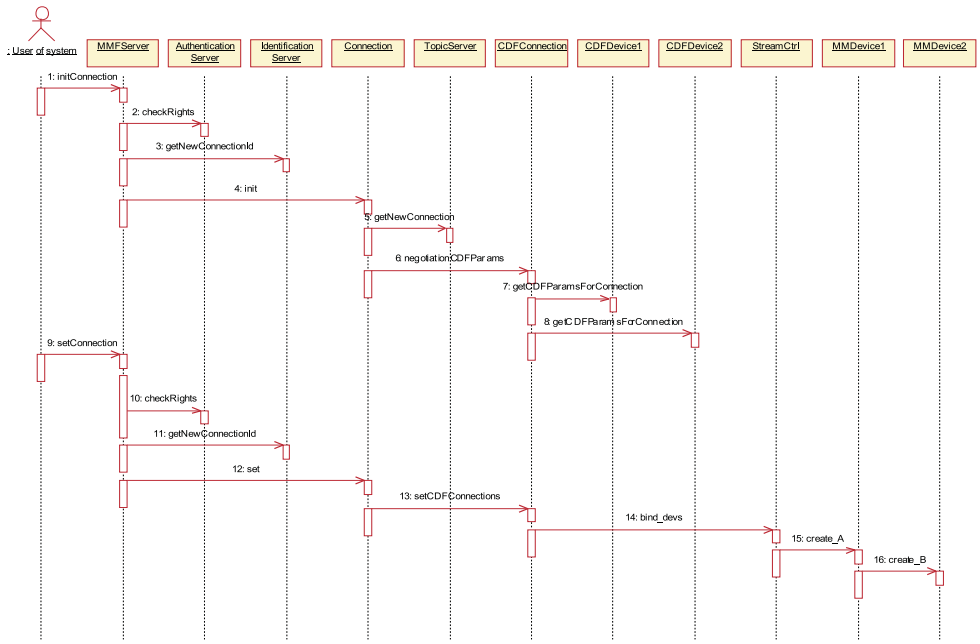

Fig. 2. UML sequence diagram describing how a new connection is established

5 Procedure of MMFramework extension with a new device

The MMFramework architecture makes it possible to easily add a new multimedia device. The procedure described below presents in four basic steps what to do to make a new network multimedia device accessible in the MMFramework system.

1. The functionality of that device should be accessed by CORBA interfaces. The role of the MMHardware CORBA Server Layer is to represent a multimedia device as CORBA object.
2. The next step is to create the MMDevice object. This component abstracts the behaviour of a multimedia device. The parameters of the object from the MMHardware CORBA Server Layer should be translated to QoS parameters which are understood by A/V Streams Control Layer. The QoS parameters give the uniform mechanisms for cooperation and negotiation of newly added device parameters' with existing components from the some layer.
3. For each multimedia device a CDFDevice object should be created. This component makes it possible to translate the parameters of the devices (such as QoS) to/from the Common Description Format (CDF). The parameters (in the CDF format) are passed by Management and Access Control Layer (MaACL). The programmer chooses which parameters of the multimedia device should be used by the MaACL and visualized by the client application.

4. The others components and items related to multimedia devices in the MaACL and the Application Layer are created dynamically by the system's administrator. All device's properties are passed to these two layers automatically, the programmer doesn't have to manually create any objects or components. These properties in a notification information form are sent from the MaACL to the Application Layer.

The last step of the described procedure, which is the creation of device's item in the Device Repository, is performed automatically by the system.

6 MMFramework client

This section describes the architecture of the MMFramework client. A graphic interface of client's program consists of two main types of components designed for it. The first component, called **NetChart**, represents a topology of the multimedia system i.e. devices and connections between them. The next components, called **Form**s, are used for presenting parameters and states of multimedia devices and connections, as it is shown in Fig. 3. For the communication of the client with the MaACL layer the programming environments such as OrbixWeb [3] and OrbixTalk [2] are used.

6.1 NetChart component

The main goal while creating the NetChart component was to achieve a compact graphic representation of the whole system, devices and connections what

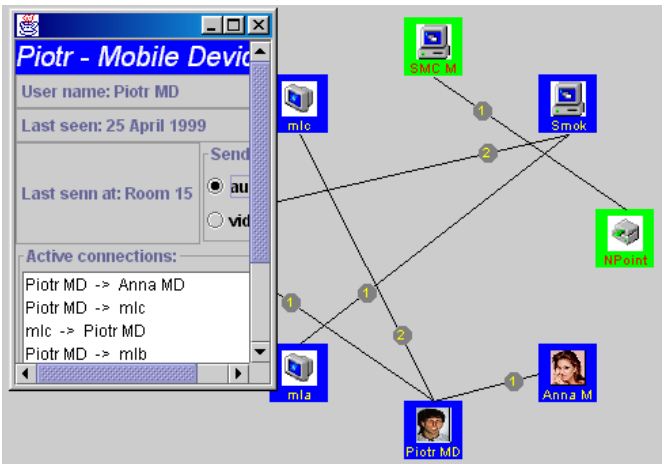

Fig. 3. NetChart component and Form component associated with Piotr MD device

provides access to their state and control functions. Thus, it allows users for a fast orientation in the system's state, making a control decisions and managing the system. Multimedia devices in this component are represented by rectangles placed on a circle as it is shown in Fig. 3. An appropriate icon and description of the device are placed inside each rectangle. Connections between devices are represented by lines drawn from one rectangle to another. Each line may indicate many connections between the same devices, therefore a label with the number of connections is plotted on each of them. This component also provides many options designed for customizing its appearance according to individual preferences. To each device and connection in the component is assigned a dedicated manager. The manager notifies them about the information changes.

6.2 Form component

The next component is a dynamically built GUI interface. It consists of simpler components called **Elem**s, which are automatically laid out on the Form in the way best fitting its size. Each elem possesses the main component which is one of six graphic Swing components such as JList, JComboBox, JLabel, JTextField, JCheckBox group and JRadioButton group [10]. These Swing components are sufficient for good representation of all possible parameters of devices and connections. Each elem has additional title and border for good appearance and also possesses its manager which delivers events connected with the elem.

The Forms are used for presentation of the parameters of devices and connections or can be edited to introduce new values of parameters. These values are used for setting devices and connections parameters. The example of graphical representation of the Form is shown in Fig. 3.

6.3 Client communication mechanism

A communication from the client to the MMF Server is done using the IIOP protocol. General assumption is that the items from the MaACL layer communicate with the client using one-way multicast message propagation mechanism used by OrbixTalk. A communication from the server to the client is more sophisticated because there is no an appropriate OrbixTalk implementation for the Java platform. Therefore the client program has a sublayer in the form of a server written in C++ language, which receives multicast messages from the MMF Server and propagates them to the client using IIOP protocol. This is schematically presented in Fig. 4.

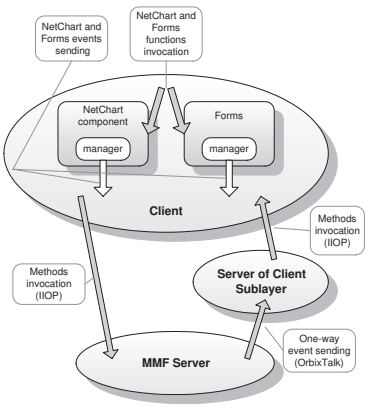

Fig.4. The communication mechanism

7 Evaluation of the system status

All layers of the architecture have been designed and successfully implemented. Of course, the system still needs further refinement, testing and validation; its basic functionality has been reached, however.

Even in such a small multimedia system as the one existing in the DCS at the UMM, it was purposeful to enrich its functionality with a scalable and efficient mechanism of notification about changes in the system's state allowing at the same time to monitor and archive events in the system.

One of the most important advantages of the MMFramework architecture is the implementation of the worked out CDF format. This powerful mechanism makes it possible to dynamically build applications. Taking advantage of CORBA architecture it has been possible to integrate different systems into one working framework.

8 Conclusions

MMFramework presents a novel approach to the multimedia applications design and implementation built around observable objects paradigm with event-based notification a principal communication mechanism. Based on the powerful OMG Control and Management A/V streams model, it unifies traditionally separate applications such as Video on demand, Videoconference and Live TV & video into one system. This is possible thanks to flexibility of the object oriented distributed programming provided by CORBA, which introduces clear conceptual model of the systems, mechanisms of dynamic binding and services. These features help to overcome inherent complexity of the multimedia systems control and management.

References

1. Continuous Media Toolkit, `http://bmrc.berkeley.edu/projects/cmt`
2. Iona Technologies Ltd., OrbixTalk White Paper, April 1996, `http://www.iona.com`
3. Iona Technologies Ltd., OrbixWeb Programmer's Guide `http://www.iona.com`
4. Multiparty Multimedia Session Control, `http://www.ietf.org/`
5. Notification Service, Joint Revised Submission by NEC System Laboratory et al., 1997.
6. Object Management Group, Control and Management of A/V Streams, OMG Document telecom/97-05-07 ed., October 1997.
7. Object Management Group, Control and Management of A/V Streams, Revised Submission, OMG Document telecom/98-10-05 ed., October 1998.
8. Object Management Group, CORBA Components, Joint Initial Submission by IONA Technologies et al., 1997, November, OMG Report.
9. SUN Microsystems, Java Media Framework, `http://www.sun.com/nab/java.framework.html`
10. Sun Microsystems Swing-1.0.1, `http://www.javasoft.com/`
11. Stream-oriented Information Flow based on CORBA Services in Large Distributed Systems, ESPRIT Project 20288-CRIT-2, March 1997
12. T. J.Mowbray and R. C.Malveau, CORBA Design Patterns, Willey, 1997.