An Extension to the CORBA Audio/Video Streaming Service: A QoS Adaptive Middleware

Antonio-Javier Garcia-Sanchez, Felipe Garcia-Sanchez and Joan Garcia-Haro

Abstract—The CORBA Audio/Video (A/V) Streaming Service has been designed to implement and integrate open distributed multimedia applications. In this paper, a set of impairments resulting of the original A/V service are identified. To overcome such limitations, we propose an extension to the specification of the A/V Streaming Service. It defines a flexible architecture creating a framework that simplifies the implementation of applications handling audio/video flows. To evaluate our service extension we develop middleware objects of an application to control and manage uncompressed video data flows with strict time restrictions. Empirical results obtained by executing this application are also presented and discussed.

Index Terms—middleware, video streaming, QoS

I. INTRODUCTION

The design of reusable components based on open distributed object computing (DOC) middleware to integrate multimedia applications is a common programming technique. This middleware governs the communication between components. Diverse middleware technologies have been proposed, being CORBA (Common Object Request Broker Architecture) one the most widely deployed standards for distributed programming.

Regarding the traditional distributed applications based on the request-reply paradigm, the design of multimedia distributed applications adds additional requirements. This is due to efficiently combine multimedia data flows transmission between remote objects along with control information associated to these flows. As a result, the OMG (Object Management Group) organization proposed the A/V Streaming Service specification [1] for CORBA.

The CORBA A/V Streaming Service facilitates the video flows transmission in real-time (if necessary) and offers a platform to create, send and receive audio and video flows between two or more generic devices (PC's, PDA's, etc.).

The CORBA A/V Streaming Service allows the separation between (1) control information of the multimedia flows where the related control data are sent by means of traditional CORBA calls and (2) multimedia data flows where the information is directly sent by using conventional transport protocols (UDP, RTP, etc.). This separation in two types of data communication confers the flexibility and portability of CORBA due to: (a) the use of CORBA layers during the connection and communication control makes possible the interoperability among applications and (b) the utilization of specific multimedia protocols provides the required efficiency in the transmission/reception of flows that are not processed by the middleware.

However, the A/V Streaming Service exhibits some limitations:

- The A/V Streaming specification allows the runtime modification of different Quality of Service (QoS) parameters associated to a flow. However, these parameters do not include some key others such as flows configuration (i.e. type of data transmission, compression, resolution, images size, etc.). To modify them, the connection has to finish and set up again for the new parameters, which implies a high, sometimes unacceptable, delay.

- Some multimedia applications require a massive interchange of control messages among distributed objects. In many cases, the transmission of these messages throughout the middleware (and the use of Internet Inter-ORB Protocol (IIOP) headers) increases unnecessarily complexity and processing time. To illustrate this effect we can suppose a local environment (i.e., a chemical industry), where several sensors act upon the image events acquired from a surveillance camera. The possible application handling these events must integrate video flows, conventional control messages, and transfer a potentially considerable number of short control messages from and towards the sensors. It would be more efficient to send these control messages by means of transport protocols as UDP (User Datagram Protocol), incurring less overhead. This cannot be done employing the current A/V Streaming Service, since all control messages must be processed by the middleware. Only the audio/video flows can be transferred by the transport protocol of choice. Therefore, an extension of the A/V Streaming Service allowing applications a greater control on the defined flows will be of developers’ interest. Moreover, the door to implement an application integrated within CORBA satisfying a wide range of performance requirements is opened.

According to current A/V Streaming version, multimedia applications requiring the above mentioned functionalities, must implement them separately, which implies an increase of the development time. In this paper, we propose an extension of the A/V Streaming Service offering the new functionalities. This extended service is implemented based on the A/V Streaming Service provided by the ACE/TAO ORB distribution [2]. Furthermore, the extended service we
propose, provides an integrated interface designed such that any application developed with the traditional A/V service is totally compatible with the new service.

To carry out a comparative performance study of the new service, two implementations of the same multimedia application have been programmed: (1) an implementation using the original/traditional TAO A/V Streaming Service, and (2) an implementation with the developed extension. This application transmits uncompressed video in multicast mode from a capture machine, towards several users who reproduce it on their screen. This application has been selected because of its intensive resources utilization. This type of applications appears in certain critical environments where the compression of video (even without losses) is not admitted. Examples of such scenarios are medical and military compression of video (even without losses) is not admitted. These applications must suitably reproduce this video flow on the user screen.

The performance figures of the CORBA A/V Streaming Service found in the specialized literature, usually evaluate QoS features like throughput, jitter, latency and CPU load [5], [6]. In this paper, an empirical study of latency and jitter at application level as a function of the CPU and network traffic load of the video capture machine is performed, and the obtained results analyzed and discussed.

The rest of paper is organized as follows: Section 2 is dedicated to the previous works. In Section 3, the current CORBA A/V Streaming Service and its extension are presented. Section 4 describes the application developed. Section 5 shows and discusses the evaluation results. Finally, section 6 concludes.

II. RELATED WORK

This section outlines works that allow to implement multimedia applications with QoS constraints supported by OMG’s CORBA and IIOP protocol. DIMMA [7] and GOPI [8] projects are middleware facilities to construct distributed multimedia applications with QoS real-time features. These middleware, compatible with CORBA, follow the A/V Service concept and have the same data separation model (control information and multimedia data flow). The implementation of DIMMA and GOPI does not follow the rules of the A/V Service specification and do not provide all its capabilities [9]. UbiQoS [10] is a middleware implemented using Java-based mobile agents to exploit Video-on-Demand services with dynamic QoS adaptation. TOAST [11] middleware platform provides component-oriented CORBA support for adaptive distributed multimedia applications, adjusting QoS parameters, i.e. to vary the achievable latency and throughput. The applications using UbiQoS and TOAST are related to mobile computing and wireless network technologies which transmit/receive compressed video.

As far as authors know, there are two works that combine the A/V Streaming Service specification and QoS features: (i) the Campus TV [5] is a tool implementing a television studio application consisting of a number of cameras and receivers over A/V Streaming Service, using a monitoring tool that computes the jitter and latency between images and, (ii) the QuO framework [6] that is designed as an intermediate layer between the application and the A/V Streaming Service and allows to implement distributed applications with QoS requirements. Its goal is to automate the mapping and translation of the application flows QoS requirements onto the QoS parameters defined by the CORBA standard. QuO introduces more complexity in the application to be implemented and is only focused on QoS issues.

III. THE EXTENDED A/V STREAMING SERVICE

A. Overview of the Original TAO A/V Streaming Service

Fig. 1 shows the original A/V Streaming Service, based on its specification [1], allows a continuous transfer of flows between two multimedia devices. In this service, each one of these flows (audio, video, data) is terminated by a flow endpoint. Also, it is possible to establish a set of flows, constituting a stream, and terminate it by a stream endpoint. For instance, in the case of two flow endpoints, we can configure one flow endpoint as a source of the data and the other flow endpoint as a sink. Note that this architecture is symmetric, allowing flow endpoints in a stream endpoint act as sources and other flow endpoints in the same stream endpoint act as sinks.

As an example, the steps describing the connection establishment of a video flow between two applications, using the original TAO A/V Streaming Service are as follows:

1. The Sender application controls the video capture of a camera, generating the video flow. The Receiver application must suitably reproduce this video flow on the user screen.
2. The Receiver application registers in the Name Service of CORBA. The Sender application invokes the method resolve of the Name Service, to obtain a reference to the Receiver object.
3. The Sender application uses the aforementioned reference to invoke the StreamCtrl object of the A/V Streaming Service. This object is inherited from the TAO_Basic_StreamCtrl object that allows the control of the data flow (start/stop the flow, etc.).

The StreamCtrl object generates two MMDevice objects, one referring to the Sender and the other to the Receiver. The MMDevice objects in their turn create two objects: (i) the
A VDev object encapsulates specific parameters of the sources, i.e., type of video format (MPEG, Raw Video, etc.) and (ii) the EndPoint object encapsulates specific transport parameters, i.e., a flow using UDP has a hostname and a port that identifies the corresponding EndPoint. A deeper explanation of the mechanism can be found in [9].

The design of classes has to be symmetrical, for both, the left part of Fig. 2 and the right part of Fig. 2. The goal is to allow the programmer to provide functionalities not supported in the original service. For instance:

1. To implement an effective flow monitoring tool and to gather statistics.
2. To modify in real-time the configuration flow parameters (no modifiable in the original standard).
3. To use the flows to exchange control messages that do not have to cross the middleware layer.

The flow connection establishment process in the extended service is similar to the original one. The TAO Basic StreamCtrl object provides access to (1) the aStreamCtrl object that provides the usual functionality of A/V Streaming Service, (2) the object joint that offers the new functionality.

The object joint allows creating the aparamoperation and bparamoperation objects. These objects are factories named endpoint_factory. Their task is to instance the final objects that implement the specific application functionality (objects: Object1... n in Fig. 2). Notice that these objects are developed by the designer of the applications.

C. New Components Implementation

To understand the modifications introduced in the traditional A/V Streaming Service, we implement a new object that is later used in the application described in Section IV. This object is named parameter_operation. The steps are as follows (see Fig. 3):

1. The joint object “bind” the paramoperation objects. The application invokes the join_devs function of joint object receiving as arguments the references to the aparamoperation and bparamoperation objects. These objects are factories that in turn create the objects designed by the programmer (object1, object2...).
2. Creation of the “objects”. Join_devs calls the creation_oper_A and creation_oper_B (1) functions in the aparamoperation and bparamoperation objects. The developer must add the object that he wants to implement (in our case parameter_operation) as an output argument in these calls. In addition, these functions further call to creation_oper_A_B (2) that distinguishes in its arguments between local and remote object.
3. Process to create a reference to the parameter_operation object in the endpoint_strategy of the A/V Service. Creation_oper_A_B invokes the creation_es function (3) of the endpoint_strategy that has as argument the parameter_operation reference (first is initialized to NULL and then, takes the corresponding value when the functions of the endpoint_strategy are executed and it returns the creation_oper_A_B (7)). Furthermore, the endpoint_strategy is formed by the activation_oper (4), acti_parameter_operation (5) and get_parameter_operation (6) functions, that obtain as result the parameter_operation reference searched and activate it in the POA (Portable Adapter of Objects).

The user invokes the functions of the parameter_operation object. Once these operations are accomplished, creation_oper_A_B returns to the creation_oper_A and
creation_oper_B functions the parameter_operation_A and parameter_operation_B (8) references. These references allow to the functions of the parameter_operation object may be invoked as local ones.

Summarizing, to incorporate an object using the extended A/V Streaming Service, the programmer must add as an out argument the object that he wants to develop in the function join_devs of the joint object. In our case, this object is denoted as parameter_operation. Then, it calls to the endpoint_factory factories (objects aparamoperation and bparamoperation) which create a remote and a local reference to the parameter_operation object and activate it in the POA. After this, the programmer may invoke to the local and remote objects.

IV. APPLICATION OF THE EXTENDED A/V STREAMING SERVICE

This section describes the application developed to evaluate the performance of our proposal, including the middleware objects of the new A/V Service. As shown in Fig. 4, the application consists of the video capture that acts as a video server and the flow transmission, in multicast mode, to an arbitrary and variable number of receiving stations that reproduce the captured video. Also, edge users can monitor the parameters of the received flow, sending towards the video capture server the monitored information about the transmission state together to other commands, as control of zoom lens, or pan&tilt movements. The working scenario is a 100 Mbps Local Area Network, with an arbitrary attached high number of PCs.

The video format is uncompressed raw video. Therefore, the volume of video data sent to the network is high (a standard 320×240 image sends to the network 230400 bytes). An image consists of a frame (video data) plus control information. Each frame is divided into 24 packets of 9600 bytes, that are injected to the network. A general-purpose Fast-Ethernet network is employed.

A. Architecture of the complete application

Fig. 4 illustrates the developed architecture. The host implementing the server camera captures high-resolution video (Sender). It is responsible of the following tasks:

- The transmission of images to the user hosts, with an additional packet indicating the creation time. It is also added a packet header that allows the user to recover the video synchronism in case of frame/image losses.
- The reception of time data sent from the user hosts.
- Analysis of the delay values received from the user hosts to estimate the state of system load. This measured delay, is the sum of the network introduced delay and the processing time consumed at the different software layers crossed. This last delay is very relevant when a middleware is used. Regarding the delay values, a decision is taken to reduce or to increase the rate of frames per second injected to the network, to be adapted to the network load conditions. The middleware objects allow defining the criteria to reduce or increase the frames per second rate: (1) manually introduced by an authorized user, or (2) a function for that purpose is included in the modified TAO A/V Streaming Service.

The receiving users implement monitoring functions that calculate the latency of each frame. A mechanism that computes the clock shift of both machines (Sender and Receiver), for a valid estimation of the latency, is also implemented. User hosts send to the Sender object the latency estimates in cycles of 50 samples that is, 50 delay values of their corresponding images. This value can be set by the programmer.

The details of the automatic mechanism implemented to adapt the system load of the host/network are as follows:

- The A/V Service layer in the video server generates (with the received statistical information from the user host) a distribution of delays per cycle.
- The reduction of the frames per second rate. When the distribution of delays exceeds in a 20% a configured given threshold, the system decreases the frames per second rate injected to the network. A reduction of the rate also occurs when the jitter values are a 50% greater than the average delay in three consecutive cycles. For example, in the working scenario, for a standard image size, 320×240, it has been observed that the delay values are around 22 ms (Pentium Dual Core). A connection that exceeds in a 33% these values results in image losses. Anticipating this possibility, the rate of frames/second sent/received in the network is decreased.
The increase of the frames per second rate. The system can increase the rate if during a certain period the average delay does not exceed a given threshold, and the values of the delay distribution are homogeneous. In this case, if the increase of the rate produces in the next three cycles an increase of the delay, the system will maintain the lowest frames per second rate, avoiding an oscillating behavior of the frames per second rate.

B. Implementation Details

The utilization of this modified TAO A/V Streaming Service facilitates the development of applications. To use it, the programmer incorporates and implements the `paramoperation` object of the new service (section III.C). From it, the following steps are:

- In the Sender side, a remote reference is created to the `paramoperation` object (a `paramoperation`). This reference allows the Sender to bind the Name Service.
- In the user host (Receiver), a remote reference to the `Sender paramoperation` object is obtained from the Name Service. In this Receiver, a local reference to its homologous `paramoperation` (b`paramoperation`) is extracted. With these two references, the next step is to call to `join_devs`. Once this operation is finished, the remote reference of `paramoperation_A` from the Sender and local reference of `paramoperation_B` from the Receiver are obtained. Using this remote reference, the `getparameter` function is called by `join_devs` of the Receiver employing as argument the values of the measured delays.
- The displaying application is responsible to calculate the end-to-end delay of the received images. This information is used to inform about them to the Receiver object inside the CORBA extension. The Receiver propagates this information to the CORBA Sender object invoking the `getparameter` function. Therefore, the Sender obtains the required information to adapt the rate of frames/second to the network and CPU load conditions. The scheme employed for this purpose is implemented in the `getresult` function of the `paramoperation` object. It is important to remark that both, network and processing delays are considered in the end-to-end delay values. This is relevant in the CORBA scenario, due to the extra CPU processing tasks added by the middleware and considered as a significant source of delay.

Moreover, the `endpoint_strategy` has been modified to adapt the new functionalities that the new TAO A/V Streaming Service offers. These changes are reflected in the application in the header files:

**User Host:**
```
TAO_AV_Endpoint_Reactive_Strategy_B<Receiver_StreamEndPoint,
   TAO_VDev,AV_Null_MediaCtrl,Receiver_paramoperation>
```

**Sender Host:**
```
typedef TAO_AV_Endpoint_Reactive_Strategy_A
   <Sender_StreamEndPoint, TAO_VDev, AV_Null_MediaCtrl, Sender_paramoperation>
```

V. PERFORMANCE EVALUATION

The effectiveness of the new modification/extension of the A/V Streaming Service has been evaluated. The test scenario consists of a PC with the functionality of video capture, and transmission in multicast mode to the receiving clients, located in three PCs. Moreover, some controlled distributed applications are additionally executed in the system to tune the network and CPU load. The employed machines are 1,8 Ghz Pentium Dual Core with 1 GB of RAM, under Windows XP® operating system, and connected through a 100 Mbps Ethernet network. The evaluation tests performed for the extension of the A/V Streaming Service are similar to other experiments with A/V Streaming Service [5], [6], where QoS parameters are tested intensively.

Figs. 5 and 6 show the effect of increasing the CPU load of the Sender on the delay suffered by the video flow. Two effects are observed: (i) the capture process slows down and some images may be not captured and (ii) the transmission delay of the packets associated to the image frames increases. If this delay is high, it will result in packet losses and therefore, some images in the user host will not be reproduced. These effects produce a non-continuous video flow traffic load feeding the network.

In Figs. 5, for the traditional A/V Streaming Service, the captured samples reveal that until 66% of CPU load the system behaves correctly and the images are reproduced without errors. However, from this CPU load value, the delay increases, becoming more critical at higher loads. Nevertheless, in Fig. 6, using the modified/extended TAO A/V Streaming Service, the system reacts once the 66% of CPU load is reached, decreasing the frames per second rate transmitted to the network and therefore, assuring that each captured image is sent in its right instant. In case of a higher reduction in the frames per second rate (<2 frames/sec.), the new service resolves that the video cannot be accurately reproduced and passes to transmit compressed video in MPEG format (allowed only for these extreme cases, better to have reproduction of compressed video than no images at all). Of course, the transmission of compressed video implies information loss, but at least the video reproduction can be done in real time, which is considered the most critical feature for the application.

The increase in the number of clients does not affect to the transmission and reception of video, because both prototypes transmit video in multicast mode and use UDP as transport level protocol. Therefore the loss of video packets does not imply packet retransmissions, and the scalability of the system is assured.

In addition, an increase of the network traffic load causes in the application a similar effect as the one due to the CPU load increase. The extended A/V Streaming Service is also valid to moderate the network congestion, reducing the frames per second rate sent when the network load increases. Additionally, the implemented application is independent of...
the underlying network technology, being able to transmit at higher video rates when the network has higher bandwidth available.

Due to the network traffic and the delay produced by the CPU load variation in the involved computers.

VII. REFERENCES


VIII. BIOGRAPHIES

Antonio-Javier Garcia-Sanchez received Industrial Engineering degree (Master) in 2000 by the Technical University of Cartagena (UPCT-Spain). Since 2001, he is joined to the UPCT in the Department of Information Technologies and Communications (DTIC) as assistant professor, obtaining the PhD. degree in 2005. His main research interests are in the areas of middleware architectures, streaming services, peer-to-peer applications and wireless personal area networks (WPAN).

Felipe Garcia-Sanchez received the Telecommunication Engineering degree (Master) by the Technical University of Valencia (Spain), performing the final Master Thesis in the Kassel Universität (Germany), in 2001. In the same year, he joined to the UPCT as assistant professor at the DTIC, where received the PhD. degree in 2005. His current research fields involve mobility and handoff systems, middleware services and performance evaluation of communications networks.

Joan Garcia-Haro is a Professor at the UPCT, Spain. He is author or co-author of more than 60 journal papers mainly in the fields of switching and performance evaluation. From April 2002 to December 2004 he served as EIC of the IEEE Global Communications Newsletter. He is Technical Editor of the same magazine from March 2001. He also holds an Honorable Mention for the IEEE Comm. Society Best Tutorial paper Award (1995).