A CORBA based platform as communication support for synchronous Collaborative Virtual Environment

Stéphane Louis Dit Picard, Samuel Degrande, Christophe Gransart

A CORBA BASED PLATFORM AS COMMUNICATION SUPPORT FOR SYNCHRONOUS COLLABORATIVE VIRTUAL ENVIRONMENT

Stéphane Louis Dit Picard
Université des Sciences et Technologies de Lille
Laboratoire d'Informatique Fondamentale de Lille
59655 Villeneuve d'Ascq cedex, FRANCE
louisdit@lifl.fr

Samuel Degrande
Université des Sciences et Technologies de Lille
Laboratoire d'Informatique Fondamentale de Lille
59655 Villeneuve d'Ascq cedex, FRANCE
degrande@lifl.fr

Christophe Gransart
Université des Sciences et Technologies de Lille
Laboratoire d'Informatique Fondamentale de Lille
59655 Villeneuve d'Ascq cedex, FRANCE
gransart@lifl.fr

ABSTRACT

This paper describes the use of CORBA middleware to support communication in a 3D synchronous Collaborative Virtual Environment called SPIN-3D: users interact simultaneously and work together on 3D shared objects. Shared objects are duplicated: each participant owns a copy of each shared object and our CORBA based platform allows to synchronize their state and to manage the session. Our platform supports two ways of communication: one using a remote method invocation mechanism for "one shoot" communication, and one other using streaming with CORBA for "flooding" communication.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*collaborative computing, computer-supported cooperative work*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual Reality*

General Terms

Communication Platform Design using CORBA Middleware

Keywords

CVE, multi-user virtual world, CORBA, MIOP, reliable multicast, multimedia streaming service

1. INTRODUCTION

Taking advantage of the rapid growth of the Internet and of the Virtual Reality technology, many Collaborative Virtual Environments (CVEs) emerge. CVEs place several users in a virtual world where they can interact with objects and each other. Synchronous CVEs provide users with the illusion that they work simultaneously: it involves the same notions as in real-life such as collaboration, communication and interaction. A CVE platform requirements are [9]:

- support for multiple participants: participants should not only visit the environment but also collaborate, communicate and interact with each other.
- data distribution and management: to achieve acceptable video frame rates, client-server data sharing is the most common approach. Each participant must have a local copy of each shared object: the CVE platform must provide a layer to manage and to synchronize the state of these duplicated objects.
- event notification and network communication: mechanisms must be provided to notify clients of events (these can be object events, such as an object state modification, or session management events, such as entering or leaving a session). It must provide efficient communication protocols to avoid network lag and to maintain the interactive feeling in the shared virtual space.
- scalability: multiple users are able to join the collaborative session without seriously degrading the quality of service.

In a multi-user virtual world, objects are duplicated on several computers. These objects are characterized by fields such as color, position, etc. An object is shared through the sharing of its characteristic fields. Previous CVE platforms such as MASSIVE [5] or DIVE [2] used their own communication layer for the events notification. The newest Common Object Request Broker Architecture (CORBA) [11] specifications provide several capabilities to support the development of Virtual Environments. Some approaches [15] use the *event service* as a method to deliver changes to several clients. The mechanism of observer-observable allows

to keep a client-server architecture, the server manages the virtual environment and notify the clients of any change. The main problem of such a solution is scalability: performance decreases when the number of clients increases. A solution to solve this problem is to sub-divide the virtual space [15]. As a client/server architecture introduces network lag, it is a problem for the interactive feeling: in a CVE, the interaction must be in real-time in order to not disturb user during collaborative action. Our approach is to use multicast communication for scalability, and multicast real-time streaming in order to keep the interactive feeling. As the communications are made in multicast, our platform gets rid of a centralized server.

2. THE DESIGN OF OUR PLATFORM

Our platform involves several services, e.g. a *lock service* in order to ensure that, at the same time, only one user has the access to an object, or a *dynamic loading service* in order to connect new objects during the session. In this paper, we present only the multicast CORBA based communication layer. We have duplicated objects onto several computers: we must synchronize their state. Our platform provide two ways for updating the remote copies of objects: the first for a simple update *(i.e. one shoot modification)*, this is ensured by the use of a remote method invocation mechanism; the second for several updates in a short time *(i.e. streaming communication)*: this is ensured by the use of multicast streams set by a stream management service.

2.1 Multicast ORB Mechanism

The CORBA standard [11] proposed by the Object Management Group (OMG) specifies a protocol, called General Inter-ORB Protocol (GIOP), to describe messages. This high level protocol defines: a Common Data Representation (CDR) to map IDL types to network representation, an Interoperable Object Reference (IOR) to point to an object and a message format (request, reply, etc). CORBA needs a transport layer to carry GIOP messages. So the OMG also specifies a unicast transport layer called IIOP (Internet Inter-ORB Protocol) which uses TCP/IP. In IIOP, each method invocation request is sent to exactly one server: an IIOP reference points only one object.

2.1.1 Group Communications

In order to synchronize the state of an object with those of its remote copies, we must be able to designate each copy. As it shown in figure 1, to allow group communication with IIOP, we must know the reference of each remote copy. However CORBA, through a multicast transport layer called Multicast Inter-ORB Protocol (MIOP) [4, 12], allows to call methods on several "identical" objects in one go: with MIOP, the reference is a group reference, so it means that several objects have the same reference. Each method call on a multicast reference is executed on each computer which has in memory an object with that reference. Currently, as it uses UDP/IP, the MIOP communication layer (as specified by the OMG in its RFP [12]) is designed for unreliable communications and supports only *oneway operations*, i.e. there is no return value and nor out parameters. As we must keep the consistency between the CVEs (i.e. the state of each shared object), we must provide a reliable communication mechanism within MIOP.

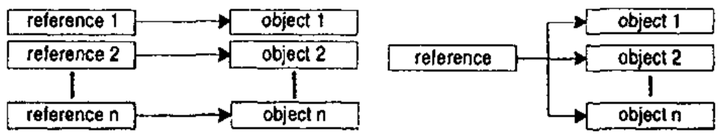

Figure 1: The notion of group with IIOP on the left side, and with MIOP on the right side.

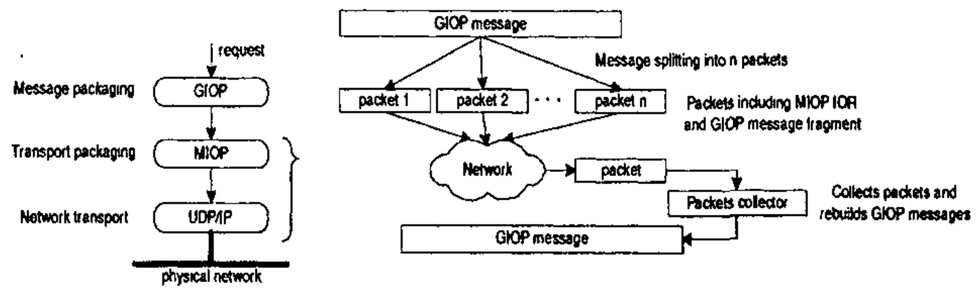

Figure 2: On the left the multicast communication stack, and on the right the GIOP message splitting.

2.1.2 Reliable Communications

At the transport level, a method invocation is one GIOP message of any size which must be sent to objects using the multicast reference. As it is shown in figure 2, each GIOP message is split into several fragments for the transport: we need to provide a reliable protocol which ensures that all connected users receive all messages, namely, all packets of all messages.

A first approach for reliable multicast is *"the sender re-emits periodically each packet until receiving acknowledgement messages from all receivers"*: this is an ACK (Acknowledgement) based protocol [8]. Such protocols can be a solution to our problem but they create significant network traffic.

A second approach is *"the sender numbers each sent packet with a unique number (incremental number), receivers detect packets lost using the current received packet number and the last received packet number, and a message is sent to the sender in order to re-emit missing packets"*: this is a NACK (Non Acknowledgement) based protocol [7]. Such protocols need an infinite flow of packets which is not our case: we have a finished number of packets for each GIOP message. We must receive all packets in order to reconstruct all GIOP messages, and there is no way to detect the lost of the last packet.

As we prefer NACK based protocols to ACK based protocols for network traffic reasons, we adjust a NACK protocol to support a limited number of packets. Each connected computer must know the number of packets of each message: this number is sent in the last packet, and this special packet uses an ACK policy in order to ensure its reception. The other remaining packets are sent using a NACK policy. Of course as it uses ACK policy for the last packet, we need to know all connected users, using a *group service* and a distributed algorithm. This work is not presented in this paper. As we are in a small group context, we do not need optimizations such as hierarchy trees [8].

2.1.3 Performances

To benchmark our Reliable MIOP (RMIOP) communication layer, we have made some experiments:

- a file transfer between two computers using different network bandwidths. It emphasizes that our multicast

transport layer over UDP/IP is as fast as TCP/IP for one server and one client. When the number of clients is multiplied by n, the global receipt time (from the sender point of view) of the TCP/IP transport layer is multiplied by n whereas the global receipt time of reliable multicast transport layer is approximately the same.

- a simple application where users can interactively change the orientation of a 3D object over network. It emphasizes the limitations of CORBA for animation: each orientation modification corresponds to one remote me-thod call, so it creates a great network traffic. The number of layers between network layer (object skeleton) and application layer (object implementation) creates problems for real-time interaction.

2.2 The Multimedia Streams Management Service over a Multicast ORB

As it is emphasized by our experiments, CORBA is unsuitable for flooding communication such as remote real-time animations. A solution is to set multicast real-time streams between duplicated objects: CORBA is a way to establish these streams. Our proposal is based on the Audio/Video Streaming service [10, 13]: this OMG standard defines a framework for the development of interoperable distributed multimedia streaming applications. The key components are the stream control protocol and the interface of the stream control objects. A flow is considered as a continuous media transfer between multimedia devices. These flows can be grouped together to form a stream. These streams are terminated by stream endpoints that contain the flow endpoints. Virtual devices (VDev) consume or produce stream data: they are a high level abstraction of multimedia devices. The stream controller (StreamCtrl) is the key component in the stream establishment protocol: it allows to bind the VDev together. The standard provides flexible support for differents kinds of multimedia services by offering a Light and Full Profile specification: this offers to application developers two levels of streams management. For the Light profile only the StreamEndpoint and StreamCtrl interfaces are exposed, whereas for the Full profile, a greater degree of control is provided as each flow interface within a stream is exposed.

We adjust the standard to our distributed and multicast architecture: we use the RMIOP communication layer in order to set the streams in one go. As we are in a multicast context, only *oneway operations* are supported: so the negotiation phase, as proposed by the standard, between the virtual devices disappears. We are in the case of the Light Profile: only the stream endpoints and the stream controller expose their interface. Each copy of a shared object is associated to a stream endpoint: a shared object implements the VDev interface, and each stream endpoint can be associated with several shared objects. Stream endpoints are duplicated, they represent a *unique logical stream endpoint*: they are implemented with a multicast CORBA object. As it is shown in figure 3, when a shared object estimates that updates need to be done through a stream, the requester object asks the stream controller to set a stream: it means that each object copy must be connected to its associated stream endpoint; this is made by the stream controller through a multicast remote method invocation on the group of stream

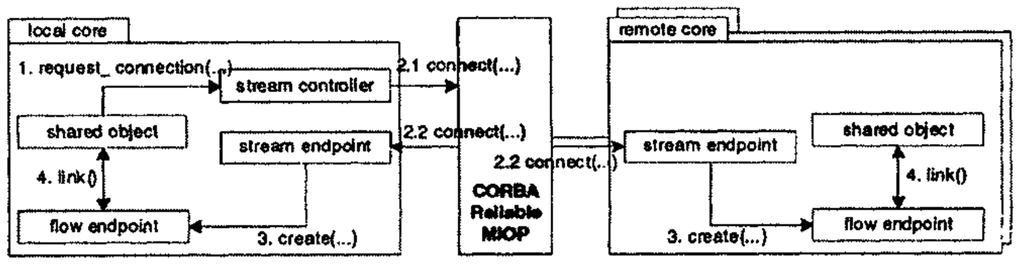

Figure 3: The multicast stream setting protocol.

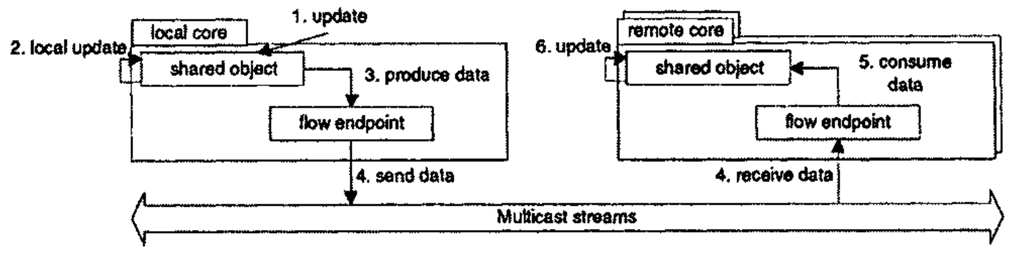

Figure 4: The multicast stream communication.

endpoints. The connection of the shared object to its respective stream endpoint creates a flow endpoint: as it is shown in figure 4, data are consumed or produced by the shared object through this flow endpoint.

So the stream is set and data transmission can start. The streams management is made by the stream controller: as the stream endpoints are multicast CORBA objects, all operations (creation, destruction, etc) are made in one go. The stream controller can be implemented as a centralized service running on a a server, to respect the architecture provided by the standard. Compared to the issues in multipoint streams provided by the standard, the MIOP communication layer enables us to control all the stream endpoints in only one go. As our platform is fully distributed (i.e. there is no centralized server), we are studying the distribution of the stream controller.

2.3 Implementation

We use the Orbacus ORB middleware [14] from Object Oriented Concept (OOC) to implement our communication platform. This middleware offers to developers a notion of communication plug-in through a framework called Open Communication Interface (OCI). Using this framework we can develop new protocols such as a MIOP communication layer. The reliable multicast protocol is implemented using C++ language over UDP/IP and is integrated into the MIOP plug-in as an extension which we call RMIOP (Reliable-MIOP): it is possible to switch dynamically between transfer policies, reliable (RMIOP) or not (MIOP). With our implementation, it is also possible to use IIOP in order to point a unique object.

The Audio/Video Streams service is also implemented in C++ and uses the RMIOP communication layer for streams management. The implementation provides a framework to easily develop multimedia applications.

3. THE SPIN-3D CVE

SPIN-3D [3] is a 3D user interface for synchronous collaborative work. Our interface is designed for small group meetings such as distant learning or co-design situations. As it is shown in figure 5, we propose a new spatial organization taking advantage of the third dimension and a 3D interaction model using two devices, one for designation and the other for manipulation of objects. To support collaboration

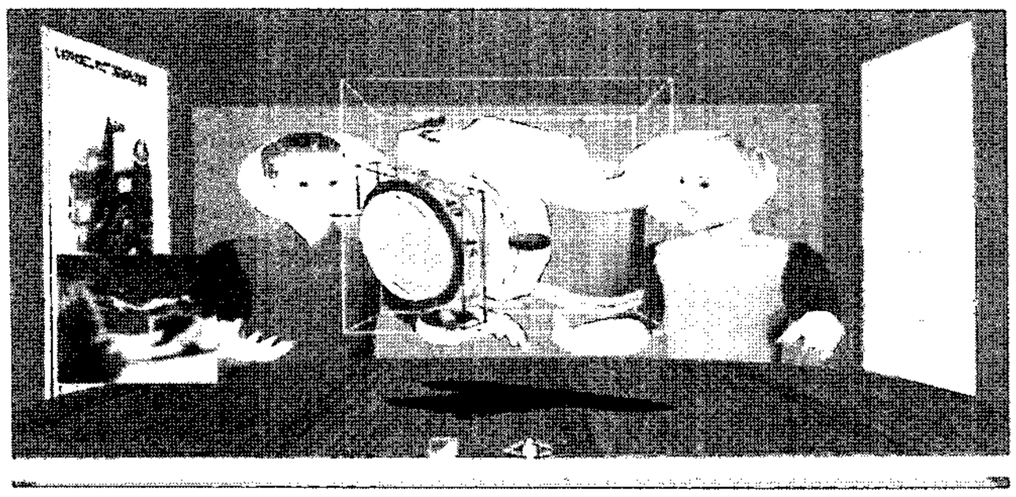

Figure 5: A learning situation in SPIN-3D.

awareness, each user is represented on remote SPIN-3D by a clone, a 3D representation of her/himself. We adopt a "conference table" metaphor: users are located around a table in a single virtual room. SPIN-3D uses the CORBA based platform described above. It uses the Virtual Reality Modeling Language (VRML) to describe 3D objects [1]: a VRML object is represented by several fields, and each field characterizes the object, for example a light is characterized by its color and its position. We improve the VRML with a multi-user extension inspired by the work of the Living Worlds group [6]. Of the implementation point of view, for a shared VRML object, each of its fields is represented by a CORBA object. The synchronization of the value of these fields is made by our communication platform: when a change occurs on a shared field (i.e. a CORBA object), the communication platform notifies its remote copies using one shoot update (i.e. using remote method invocation mechanism) or multiple updates (i.e. using the streaming service mechanism). The platform chooses the way of notification by using information contained in the VRML sharing description: for each shared field, there is an attribute in order to specify how the remote updates are made, the value of such an attribute can be "action" or "animation". So this is transparent for the SPIN-3D programmers: they only specify this information in the VRML sharing description; moreover they can modify the notification type dynamically by changing this attribute without any call to the communication platform. This is useful for objects animations: an object is remotely animated using streaming. Control operations on streams (such as creation, destruction, etc) and data transport are made transparently by the communication platform.

4. CONCLUSIONS

This paper presents the design of a CORBA based communication platform for synchronous CVE. It allows to share data among several computers: the synchronization is made by using multicast remote method invocation and multicast real-time streams. CORBA and its newest specifications such as MIOP and Audio/Streaming service provide a flexible and efficient solution to many of the technical issues in Virtual Environment platform. The aim of SPIN-3D is to provide a platform with which it will be easy to develop synchronous collaborative applications.

5. ACKNOWLEDGEMENTS

The research project reported here is supported by France Télécom R&D (FT R&D) and the regional council of Nord-Pas de Calais (FRANCE).

6. ADDITIONAL AUTHORS

Christophe Chaillou (USTL-LIFL) and Grégory Saugis (FT R&D, email: `gregory.saugis@francetelecom.fr`)

7. REFERENCES

[1] R. Carey, G. Bell, and C. Marrin. Iso/iec 14772-1:1997 virtual reality modeling language (vrml97). www.vrml.org/Specifications/VRML97/.

[2] O. Carlsson and O. Hagsand. Dive - a platform for multi-user virtual environments. *Computer and Graphics*, 17(6):663–669, 1993.

[3] C. Dumas, S. Degrande, G. Saugis, C. Chaillou, M.-L. Viaud, and P. Plénacoste. Spin: a 3d interface for cooperative work. *Virtual Reality Society Journal*, 1999.

[4] C. Gransart and J. M. Geib. Using an orb with multicast ip. In *PCS'99 Parallel Computing Systems Conference Proceedings*, Ensenada, Mexico, August 1999.

[5] C. Greenhalgh and S. Benford. Massive, a collaborative virtual environment for tele-conferencing. *ACM Transaction on Computer Human Interaction*, 2(3):239–261, September 1995.

[6] Living World Working Group. Making vrml 97 applications interpersonal and interoperable. www.vrml.org/workinggroups/living-worlds/.

[7] T. Liao. Light-weight reliable multicast protocol specification. Internet-Draft, draft-liao-lrmp-00.txt, webcanal.inria.fr/lrmp/index.html, October 1998.

[8] J. C. Lin and S. Paul. Rmtp: a reliable multicast transport protocol. In *IEEE INFOCOM'96 Conference Proceedings*, pages 1414–1424, March 1996.

[9] B. MacIntyre and S. Feiner. A distributed 3d graphics library. In *SIGGRAPH'98 Conference Proceedings*, pages 361–370, Orlando, Florida, July 1998.

[10] S. Mungee, N. Surendran, and D. Schmidt. The design and specification of a corba audio/video streaming service. In *HICSS-32 Hawaii International Conference on System Sciences Conference Proceedings*, Hawaii, January 1999.

[11] OMG. The common object request broker: Architecture and specification revision 2.3. OMG Document formal/98-12-01, June 1999.

[12] OMG. Unreliable multicast inter-orb protocol request for proposal. OMG Document orbos/99-11-14, November 1999.

[13] OMG. Control and management of audio/video streams specification. OMG Document formal/2000-01-03, January 2000.

[14] OOC. The orbacus home page. www.ooc.com/ob/.

[15] S. Wilson, H. Sayers, and M. McNeill. Using corba middleware to support the development of distributed virtual environment applications. In *WSCG'2001 Conference Proceedings*, Plzen, Czech Republic, February 2001.