Design and Assessment of a CORBA-Based Multi-tier Architecture for groupware and networked multimedia applications control

Khalil DRIRA, Saul POMARES HERNANDEZ, Michel DIAZ
Laboratoire d'Analyse et d'Architecture des Systèmes
7, Av. du colonel Roche - 31077 Toulouse - France
{ khalil, spomares, diaz} @laas.fr

Abstract

This paper describes and evaluates a CORBA-based multi-tier architecture for the design and implementation of Groupware and Networked Multimedia applications. Design is based upon a layered model separating cooperation, coordination and communication. The communication is implemented using remote method invocation through the Object Request Broker according to the CORBA standard. The evaluation shows the appropriateness of the multi-tier model to support both coordination and cooperation traffics.

Key-words*: CORBA, Groupware, Networked Multimedia Applications, Design Assesment, Multi-tier Architecture.*

1. Introduction

Groupware and networked multimedia applications are becoming very popular since the generalization of internet connectivity. The complexity of these applications is increasing with the number of services they offer and the number of interacting users who simultaneously interact with each other using the support applications. Structuring the many levels of interactions found in such applications by identifying generic services helps to achieve code reuse, with the benefits of making implementation and reasoning on the application easier, and of reducing development costs.

Today, although a large number of groupware tools exists, the modalities of sharing information are often inappropriate and limited. Hence the paradox : online cooperation creates bottlenecks rather than enhancing the productivity and interaction between participants in computer-mediated activities [Dommel HP. and Garcia Luna Aceves J.J. 1997]. Several studies have been conducted recently to design new architectures and description techniques for an appropriate design of coordination in distributed applications [Adler R.M. 1995, Lewis T.G. 1995].

We focus in this paper on CORBA-based architectures for groupware and networked multimedia applications. We develop CCML (Coordination-Centered Multi-Layers model), a general model for the design of groupware and networked multimedia applications that ensures tailorability and collaborability of applications through the generic coordination layer. Using the service of the underlying communication layer, the coordination layer provides, as a service for the upper layer, a set of customizable modalities for sharing information with fine-granularity and a service for dynamic architecture evolution management.

The layered model is implemented according to a multi-tier architecture using the middleware technology of CORBA [Mowbray Thomas J. and Zahavi Ron 1995]. The measures we undertaken shows that our multi-tier architecture is effectively functional for implementing control and data exchange of groupware and multimedia applications over high speed networks. Our model has the benefits of ensuring evolutivity and scalability using object-oriented desig. and implementation techniques.

The paper is organized as follow: Section 2 presents CCML, the multi-layers model we propose to design architecture of groupware and networked multimedia applications. Section 3 presents the multi-tier architecture supporting this model and summarizes our practical experience with this architecture. Section4 presents the assessments results of this architecture. The conclusion summarizes the benefits of the approach and suggests further work.

2. The CCML design model

The Coordination-Centered Multi-Layers model (CCML) identifies three levels of interactions (Fig.1):

The *Communication layer* provides generic services for site creation and communication. For instance, maintaining groups of sites on a network, providing multicast communication services between them, pertain to that level. Such requirements are found in many distributed applications. Further, some services at that level may be safely shared between applications, rather than duplicated (e.g. name servers).

Immediately above, the *Coordination layer* is in charge of enforcing particular interaction strategies among group members; these typically follow from constraints unknown at communication level (e.g. preserving consistency of some data shared between sites). Architecture Management, or creation/destruction of agents or of connections, also belongs to that layer.

Finally, the Cooperation layer implements the protocols the distributed sites need to achieve a common final goal. That goal could be, for instance, a complete software system, the development of which is shared among several teams according to some workplan, timetable, priorities and consistency constraints.

It is obviously harder at that level to identify generic services, but more specific layering techniques can be used to structure further that layer.

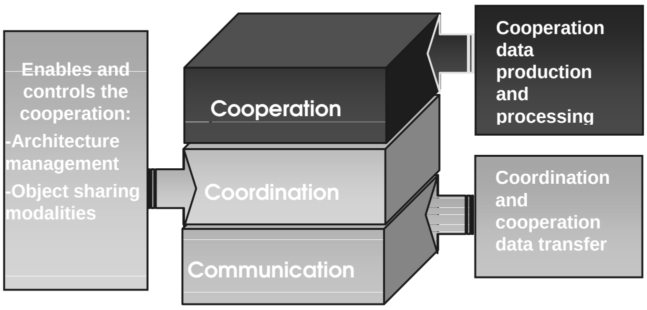


Fig. 1 The CCML design model for groupware and networked multimedia applications

Applications built according to the CCML model have an abstract architecture. Cooperating objects run concurrently on distributed *sites*, they are organized into *groups*. When no coordination action is involved, cooperating objects may directly communicate with their peers, without using the coordination medium. This is called plain intra-group cooperation. Cooperating Objects use the coordination service when communication is not sufficient. This includes architecture transformation requests allowing for example a user to access the shared data and in the same time a cooperating objects group to be created for the management of the consistence of this information between the different connected sites.

3. The multi-tier architecture

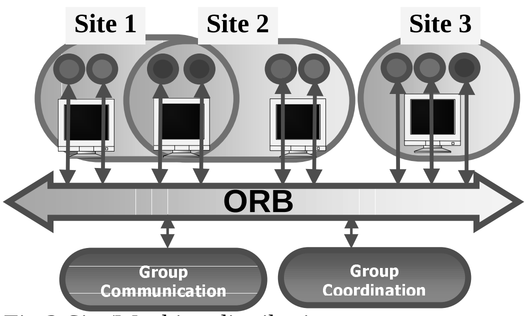


Fig.2 Site/Machine distribution

In practical utilization, a cooperating site may run on several machines, and conversely, a machine may execute objects of different sites, as illustrated by Fig. 2.

3.1 The two families of multi-tier architectures

Two multi-tier architectures are defined as shown in Fig. 3: CBMA (Communication Base Milti-Tier Architecture) is used when no coordination action is involved, cooperating objects may communicate with their peers, through the communication server (Fig. 3a). CCBMA (Coordination Communication Base Multi-Tier Architecture) in this case Cooperating Objects use the coordination server when coordination is required. Actions are then forwarded to the peers objects using the communication server (Fig. 3b).

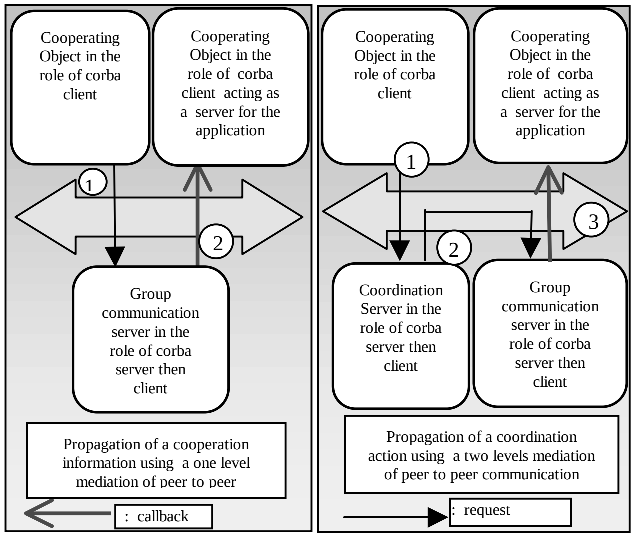


Fig.3 Communication paths and components a) CBMA b) CCBMA.

The proposed architectures have the following common characteristics:

They allow cooperating objects to invoke remote methods on their peers using callbacks initiated by the group communication server.

They allow conformance to the security requirements to be fulfilled by the WEB-based applications implemented using Java applets when the coordination and the group communication servers are hosted by the web server.

3.2 Implementation and experiments

We have experimented the multi-tier architecture presented above for the implementation of CDK, a CORBA-based Toolkit under Orbix (6800 lines, 96 classes in Java). It provides a coordination and a communication servers and other facilities to develop Cooperating objects either in Java or C++.

A formal approach for coordination protocols synthesis has been developed and applied to CORBA [Drira K., Gouëzec F. and Diaz M. 1999 a] and concurrent processes [Drira K., Berthomieu B. and Diaz M. 1999 b].

Two generic coordination models have been defined and implemented in CDK: the linear coordination model [Drira K., Gouëzec F. and Diaz M. 1999 a], and the hierarchical coordination model [Drira K., Gouëzec F. and Diaz M. 1999 b]. Both models are being integrated to describe architecture management and data sharing in inter-companies collaborative activities.

4. Assesment of the architecture

The CORBA Architecture is based on the ORB (Object Request Broker) which acts as a software bus and makes possible the invocation, by any client located on the network, of the methods of any servant object connected to the bus. The measurement process through remote method invocation involves the following stages:

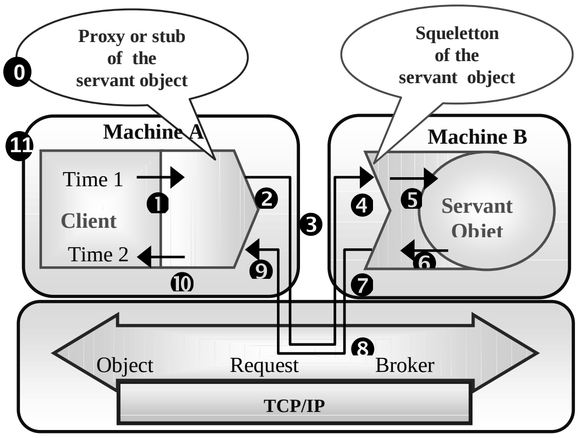


Fig 4 remote method invocation and time record.

0. Get time Time1: `gethrtime()`
1. Invocation of a method by the client .
2. Packing of arguments of request side client .
3. Transport of the request through the network .
4. Unpacking the arguments of request at the server side.
5. Invocation of method on the real object.
6. Return of the local invocation .
7. Packing of the request result.
8. Transport of the request result through the network .
9. Unpacking of the request result at the client side.
10. Return of the remote invocation and transmission of the result to the client.
11. Get time Time2: `gethrtime()`.

4.1 Generalization for the higher multi-level architectures

As illustrated in Fig.4., the CORBA communication procedure is symmetric and allows us to identify

the time taken by the middleware to transport a request from the client to the server and the response back to the same client (Fig.4),

with the time taken by the middleware to transport a request from the client to the server and a callback from this server to another client (right part of Fig.2)

5. Evaluation.

The evaluation was made on two widely used brokers (OmniORB and Orbix). OmniORB is a slight implementation of CORBA; it supports CORBA Core (ORB and partially BOA (Basic Object Adapter)) which is completely functional, but does not provide the Dynamic Invocation Interface (DII), the Dynamic Skeleton Interface (DSI), the Interface Repository (IR), nor the Implementation Repository. Orbix was the first to implement standard CORBA. One of its main characteristics is that it supports a wide range of platforms and has a great number of products that accompany it (Object Services).

5.1 Evaluation Description.

The evaluation is based on three important issues. The first is to obtain and to compare the local RTT (Round Trip Time), where the client and the server run in the same machine, and remote RTT, where the client and the server run in the same machine, for the CBMA and for CCBMA (Fig. 3). The second issue also involves the local RTT and remote RTT evaluation*;* but in this case, we increased the number of clients up to 10, using a maximum data buffer size of 1 kB. The third evaluation is undertaken in order to find the degradation in the communication when increasing the data buffer size to 1, 10, 100, and 500 kB.

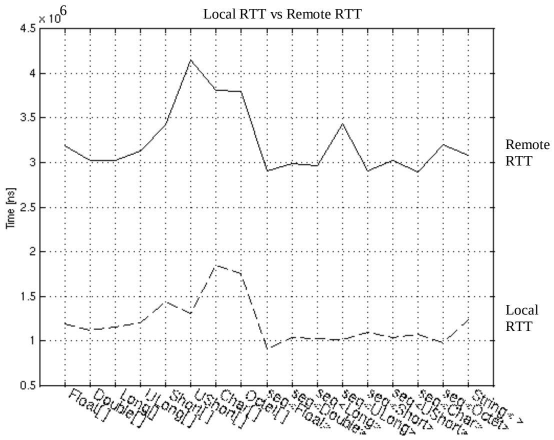
100 tests for each specific situation were made, and the presented data is an average of the obtained results. At the end of the paper we present the comparison of the results obtained from evaluated brokers. In addition**,** some conclusions are presented.

Description of the work environment:

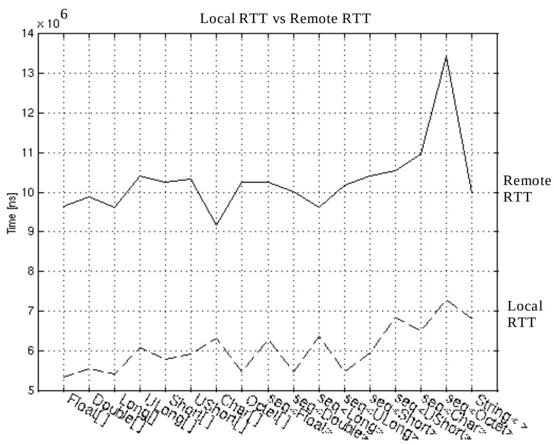
Broker	**OmniORB**	**Orbix**
Manufacturer	Olivetti y Oracle Research Laboratory	Iona Technologies PLC
Version tested	OmniORB 2.5.0	Orbix Beta 3.0
Used platform	• SunOS 5.5 • SPARCstation 20, main memory 160Mb., virtual memory 155 Mb. • SPARCstation 10, main Memory 128 Mb., virtual memory 99 Mb.	
Network	10BASE-T Ethernet.	

5.2 Using 1 client and 1 server.

Using 1 client and 1 server (Fig. 5), we observed in both brokers that the time required to send a 1 kB data buffer of local RTT form as remote RTT, differs according to the data type. This is due to the marshaling and demarshaling process. For OmniORB the remote RTT time increases an average of 256% in comparison to the local RTT time and for Orbix only it increases 116%.



a)



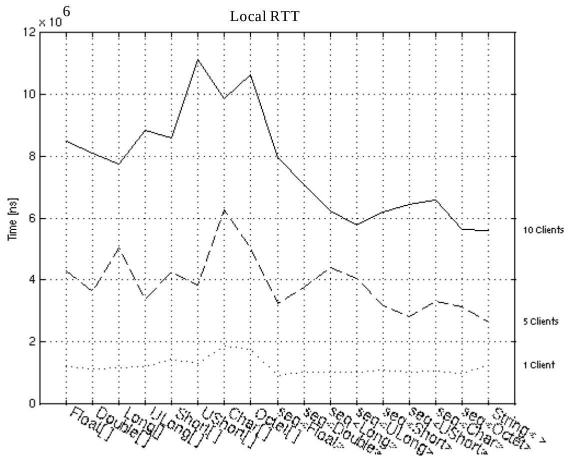
b)

Fig 5 Transfer time of 1kB, using different data types a) OmniORB, b) Orbix.

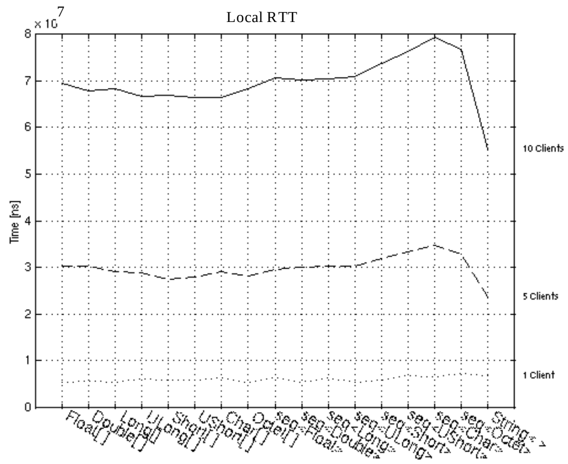
5.3 Using 1, 5, 10 clients and 1 server

As depicted in figures 6 and 7, the time of local RTT and remote RTT for both brokers increases proportionally as the number of clients increases. For OmniORB, a marked difference exists between the transmission times of the different data types as the number of clients increases. Orbix presents a homogenous increase between the data types, except in the type sequences for 10 clients.

The remote RTT time in OmniORB using 5 and 10 clients increases an average of 214% and in the case of Orbix, the remote RTT increases an average of 129% with respect to the local RTT time.



a)



b)

Fig. 6 Local RTT time of 1kB, using 1, 5 and 10 clients a) OmniORB, b) Orbix.

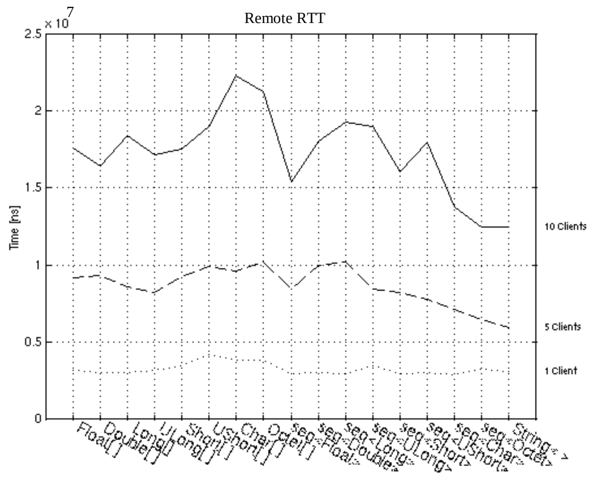
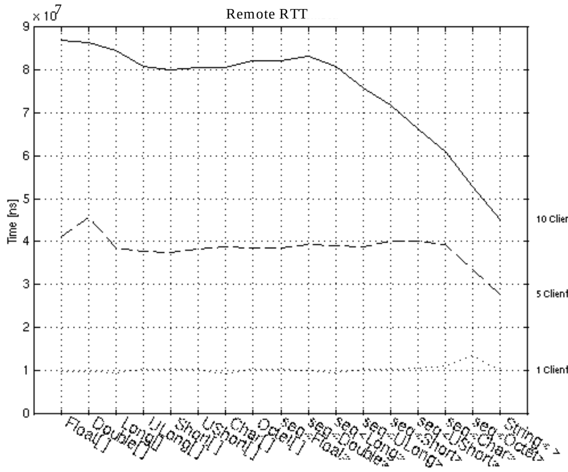


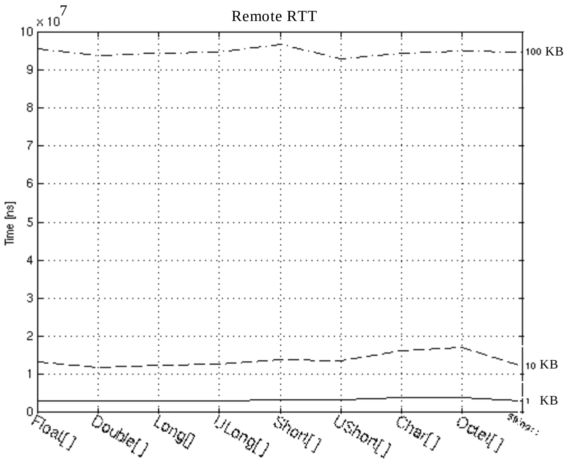
Fig 7 a) Remote RTT time of 1kB, using 1, 5 and 10 clients and using OmniORB.



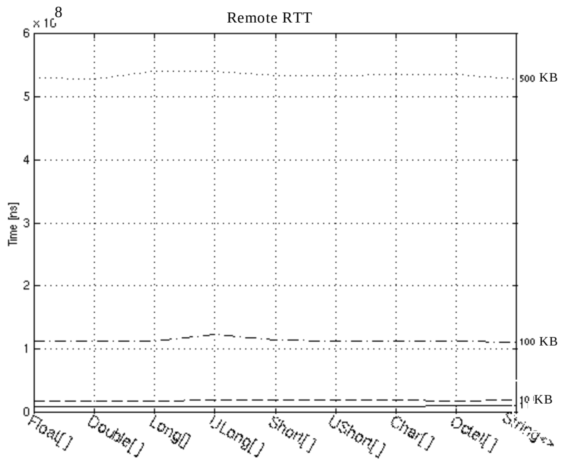
b

Fig 7 b) Remote RTT time of 1kB, using 1, 5 and 10 clients and using Orbix.

5.4 Increasing the size of the data buffer.



a)



b)

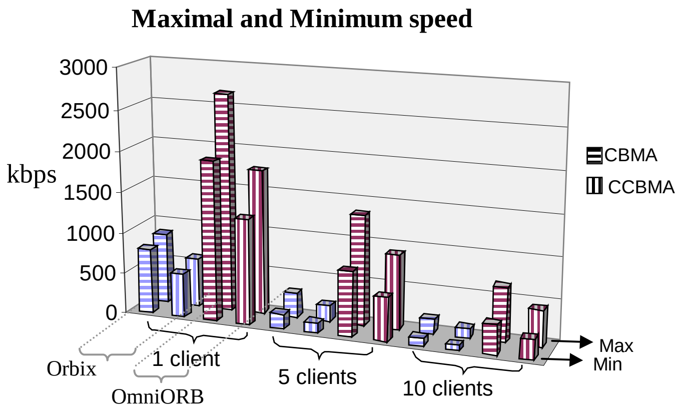
Fig. 8 Remote RTT time of 1, 10, 100, 500 kB. Using a) OmniORB, b)Orbix.

Increasing the data buffer size from 1 to 100 kB (500 kB for Orbix), we notice that for both brokers the time has increased in direct proportion to the buffer data size, displaying a regular behavior as shown in Fig. 8.

When using a buffer of 500 kB the broker OmiORB presented transmission problems.

6. Comparison of the Brokers OmniORB vs Orbix.

In this section we show some comparative tables of brokers that have been evaluated. The following graphs and illustrations show only the part of the assembly data, which we believe to be the most important.



speed	1 Client				5 Clients				10 Clients			
	Orbix		Omni		Orbix		Omni		Orbix		Omni	
	CBMA	CCBMA	CBMA	CCBMA	CBMA	CCBMA	CBMA	CCBMA	CBMA	CCBMA	CBMA	CCBMA
Max	870.4	588	2666.4	1777.6	296	197.6	1356	908.8	177.7	118.4	666.4	444
Min	800	532.8	1951.2	1289.6	177.6	118.4	800	532.8	92.8	61.6	364	242.4

Fig 9. Maximal and minimum speed in kbps using 1 client, 1 server.

The tables and graphs presented allow us to obtain the following information:

The local RTT and remote RTT times used by broker Orbix in each one of the different analyzed cases are considerably greater compared with omniORB. Which it is verified in [MLC Systeme GmbH and Charles University, 1998] and [Lo the Pope, 1998].

OmniORB didn't support, under the same conditions of work as Orbix, the transmission of 500 kB.

Data type	1 Client		5 Clients		10 Clients	
	Orbix	Omni	Orbix	Omni	Orbix	Omni
Float[ ]	0.96	0.31	4.1	0.91	8.6	1.7
Double[ ]	0.98	0.30	4.5	0.92	8.6	1.6
Long[ ]	0.96	0.30	3.8	0.85	8.6	1.8
ULong[ ]	1.0	0.31	3.7	0.81	8.0	1.7
Short[ ]	1.0	0.34	3.7	0.92	7.9	1.7
UShort[ ]	1.0	0.41	3.8	0.99	8.0	1.9
Char[ ]	0.91	0.31	3.8	0.96	8.0	2.2
Octet[ ]	1.0	0.37	3.8	1.0	8.2	2.1
String<>	1.0	0.30	2.7	0.59	4.5	1.2

Table 1.Remote RTT Time Average [ns] $x10^7$ using 1 server.

Data type	10 kB.		100 kB.		500 kB.	
	Orbix	Omni	Orbix	Omni	Orbix	Omni
Float[ ]	0.18	0.13	1.1	0.95	5.3	---
Double[ ]	0.18	0.11	1.1	0.93	5.2	---
Long[ ]	0.18	0.12	1.1	0.94	5.3	---
ULong[ ]	0.19	0.12	1.2	0.94	5.3	---
Short[ ]	0.18	0.13	1.1	0.96	5.3	---
UShort[ ]	0.18	0.13	1.1	0.93	5.3	---
Char[ ]	0.18	0.16	1.1	0.94	5.3	---
Octet[ ]	0.18	0.17	1.1	0.94	5.3	---
String<>	0.18	0.11	1.1	0.94	5.2	---

Table 2. Remote RTT Time Average [ns] $x10^8$ using 1 client, 1 server.

The maximal speed using 1 client, 1 server and the CBMA for OmniORB was 2666.4 kbps, and for Orbix was 870.4 kbps(Fig. 10).

The maximal speed using 1 client, 1 server and the CCBMA for OmniORB was 1777.6 kbps, and for Orbix was 588 kbps. (Fig. 10).

The maximal speed using 10 client, 1 server and the CBMA for OmniORB was 666.4 kbps, and for Orbix was 177.7 kbps(Fig. 10).

The maximal speed using 1 client, 1 server and the CCBMA for OmniORB was 444 kbps, and for Orbix was 118.4 kbps. (Fig. 10).

7. Conclusion

This paper described the design and evaluation of a layered architecture for distributed cooperative applications. According to the proposed architecture, applications are cast in a three-level architecture implementing separately cooperation, coordination and communication. This improves modularity, reusability, evolutivity, and cost of development. The assessment of the CORBA-based implementation for these architectural families shows that layering does not lead to degradation of efficiency and architectures designed in this manner may support both traffic control and huge cooperative data exchange.

8. Acknowledgment

This work has been partially supported by the TELECOMMUNICATION Program of CNRS under grant TL-97028, by the ECOS-NORD Program under grant M99M02 and by the IST program of the European Community under grant IST-1999-10302 within the DSE (Distributed System Engineering) project.

9. References

Adler R.M. 1995. Distributed coordination models for client/server computing. IEEE Computer, 28(4):14-22, April 1995.

Aggarwal S., Barbara B. and Meth K.Z. 1988. A software environment for the specification and analysis of problems of coordination and concurrency. IEEE Transactions on Software Engineering, 14(3):280-290, March 1988.

Dommel HP. and Garcia Luna Aceves J.J. 1997. Floor Control for multimedia conferencing and collaboration. Multimedia Systems (1997) 5:23-38.

Drira K., Gouëzec F. and Diaz M. 1999 a. Design and implementation of coordination protocols for distributed cooperating objects. A general graph-based technique applied to CORBA In the proceedings of IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS), Florence (Italy), 15-18 February 1999. FMOODS, Eds. P.Ciancarini, A.Fantechi, R.Gorrieri, 1999, Kluwer Academic Publishers, ISBN 0-7923-8429-6, pp.89-104

Drira K., Gouëzec F. and Diaz M. 1999 b. A cooperation service for CORBA objects. From the model to the applications.. In the proceedings of EUROPAR'99, Toulouse (France) September 1999. Lecture Notes in Computer Science V. 1685 Springer, pages 769-776.

Drira K., Berthomieu B., and Diaz M. 1999 An Approach to the Synthesis of Coordination in Distributed Cooperative Applications. In CSCWD99 - Computer Supported Cooperative Work in Design, September 29-October 1, Compiegne, France, 1999.

Lewis T.G. 1995 Where is client/server software headed? IEEE Computer, 28(4):49- 55, April 1995.

Mowbray Thomas J. and Zahavi Ron 1995. The Essential CORBA: Systems Integration Using Distributed Objects. OMG/Wiley (ISBN 0-471-10611-9), 1995. http: //www.omg.org/tombook.htm.

Drira K., 2000. A coordination middleware for collaborative component-oriented distributed applications. Special Issue of NETNOMICS journal on information and communication middleware, page 18pages, 2000. To appear . in July 2000

Drira K. and Diaz M. 2000 Graph-grammar based coordination in inter-corporate computer supported collaborative activities. Annual Review of Scalable Computing, 2:27pages, To appear in 2000.

Lo and Pope, 1998. Sai-Lai Lo, Steve Pope, The Implementation of a High Performance ORB over Multiple Network Transports, 1998.

MLC Systeme GmbH and Charles University, 1998. MLC (Management Logistic Communication), Charles Univesity, Prague, Csech Republic, *CORBA Comparison Proyect*, Jun 17 1998.