Extension To Corba Event Service For a Conference Control System

João Orvalho[1], Tiago Andrade[2] and Fernando Boavida[2]
Departamento de Engenharia Informática, Universidade de Coimbra - Pólo II
Communications and Telematic Services Group
3030 COIMBRA - PORTUGAL
Tel.: +351-39-790000, Fax: +351-39-701266, E-mail: orvalho@dei.uc.pt

Abstract

Conferencing applications require an efficient conference control service. The present paper introduces the main features of a conference control service based on the CORBA Event Service using the Java platform, under development at the Communications and Telematic Services Group of the University of Coimbra. This framework is based on the ITU T.120 recommendations, with reliable multicasting communication for both data and control information. The control service extends the CORBA Event Service architecture with mechanisms for reliable IP multicast communication, with total ordering of the events transmission, filtering and fragmenting/reassembling of large messages.

1. Introduction

Conference applications rely on groups of geographically dispersed linked nodes that are capable of exchanging audio-graphical and audio-visual information, through different types of communication networks.

At present, conference applications play a key role in workgroup tools. An efficient conference service is fundamental to facilitate the development and control of this type of applications. A nuclear aspect of conference applications is their ability to use multipoint communication in heterogeneous environments.

In the scope this work, a conference framework was conceived using the Java™ platform [1], based on the ITU T.120 recommendations, with reliable multicasting communication for both data and control information.

The control information is managed by a service implemented in Java, according to the CORBA Event Service model [2].

The present paper introduces and discusses the main features the proposed conference control service. In section 2, the state of the art and related work are presented. Section 3 provides a general description of the conference framework architecture, with some emphasis on reliable multicasting communication and on system synchronism. Still in this section, the specification and the implementation of the proposed framework are presented. Finally, section 4 presents some concluding remarks and lines for further work.

2. State of the art and related work

There are several proprietary architectures for this type of collaborative applications, some of which are referenced below. However, the proposal under consideration in the present paper is based on solutions recommended by international standardisation institutions (basically, ITU-T and IETF) and, thus, we will focus our attention on systems that somehow relate to these standardised solutions.

There are two basic types of conference control: tightly-coupled/formal and loosely-coupled/informal, which, some times, can be combined into hybrid control schemes.

The tightly coupled approach is based on a central and authoritative control mechanism, managing the members and the conference access. On the other hand, the loosely coupled approach uses a distributed control that does not require an explicit control mechanism for members and applications. These two control models are so different that interaction between conference applications using different control models is, in practice, ruled out.

These two approaches are proposed by two standardisation institutions: International Telecommunication Union (ITU) and Internet Engineering Task Force (IETF). Typically, a loosely coupled conference consists of multipoint sessions with audio and video streams, supported by RTP and RTCP protocols, using IP Multicasting, as it is the case of MBone [3] in the Internet community. ITU defined two standards for tightly coupled conferences: T.120 [4-12] recommendations for data conferences and H.323 recommendations [13] for multimedia conferences. The increasing interest in more formal conferences lead IETF

[1] College of Education, Polytechnic Institute of Coimbra
[2] Informatics Engineering Department of the University of Coimbra

0-7695-0253-9/99 $10.00 © 1999 IEEE

to the development (still as work in progress) of a tightly coupled model - "SCCP, Simple Conference Control Protocol" [14] - with functionalities that are similar to ITU T.120 and H.323 functionalities.

A great number of characteristics of conference applications are common to various scenarios. The use of generic functionalities simplifies the development of this type of applications, creating conditions for the use of normalised models which, in turn, makes possible the interconnection of different systems.

Generally, typical conference systems (whiteboard, audio and video) are characterised by the following functionality, in addition to security, streaming and application-specific functionalities [15]:

data transmission	unicast/multicast, data serialisation-marshalling, priorities, addressing and transparent localisation, internetworking
conference management	conference set-up and termination, database administration, conference merging and splitting, resource management, etc.
distributed applications support	synchronization, conductorship

Some of the nuclear aspects in a conference service are the multipoint communication between different users in heterogeneous environments, and the scalability directly associated to the resource control.

One of the factors that has more influence on the model scalability is the resource request response time, as it is the case for the channels and tokens. The response time can be used as a measure of the system effectiveness, and has a direct influence on the maximum number of users/systems participating in a conference and on the amount of exchanged information.

In the T.120 case, a central database in a top server is used, which can result in high response time both in terms of resource requests and information update. In the case of the Internet community, IETF bases its solution on a distributed database, obtaining smaller resource request response times. However, the IETF proposal implies a high load in the data updating in all conference nodes[16].

ITU T.120 protocol stack includes multipoint communication and generic control of tightly coupled conferences through different types of networks (ISDN, TCP/IP, etc). Conferences have a tree topology, based on a top provider node with dedicated control functions, which leads to high response times in terms of resource requests and an insufficient updating mechanism of the conference database [9] [16]. Additionally, they do not provide support for real-time multicasting, for both control information and data [17], and they have poor flexibility in terms of conference merging/splitting operations.

Surely, ITU T.120 based conference systems are more evolved, as reflected by the market of "desktop conferencing". Nevertheless, they exhibit some scalability limitations [15], which arise from the above mentioned high resource request response times, from the mechanisms used for replication/updating of conference data control and also from the lack of support of real-time reliable multicasting communication, for both data and control information.

IETF (Internet Engineering Task Force) proposes the SCCP (simple conference control protocol) [14], the SCIP (simple conference invitation protocol) [18] and multicast protocols like RTP (real-time protocol) [19]. The characteristics of the IETF protocol for tightly coupled conferences control are identical to the characteristics of ITU T.120. Unfortunately, at the moment, the description of this protocol is sparing in details, which constitutes an obstacle to its adequate evaluation [17].

CCS (CORBA-based Conference Service) is a conference control service with functionalities similar to ITU T.120, but with high scalability and support for real-time reliable multicast. This framework is based on the CORBA object model. The service is not tested and presents limitations in terms of multicasting communication, especially with the integration of OrbixTalk [20], which is an implementation of the CORBA *Event Service* specification normalised by OMG [2]. OrbixTalk uses IP multicasting and a reliability mechanism based on negative "acknowledgements", in order to provide delivery confirmation of each information object. However, it doesn't supply total ordering of multicast data, which in certain high load situations can lead to the loss of messages [21].

HORUS [22] is a distributed conference system supported on CORBA platforms, that presents a proprietary synchrony mechanism: *Virtual Synchronization* [23]. In this model the data marshalling is made by CORBA and it supports multipoint communication through a non-standard proprietary platform [17].

3. Conference System

3.1. General description of the architecture

The proposed system is based on the tightly coupled conference paradigm, and has the aim to achieve a high scalability degree. To reach such objectives we have conceived two nuclear services, one for conference control - *Conference Controller* (CC) - and another one for multipoint communication between collaborative

applications (Figure 1). The CC service, as responsible for the administration of the conference, implements a minimal set of the basic functionalities of ITU T.124 recommendation (GCC- Generic Conference Control) [8] [24] [25], and uses a reliable service for multicasting communication. For the applications data multipoint communication, we have chosen to use ***JSDT*** (Java Shared Data Toolkit) [26]. JSDT is a framework based on the ITU T.122 recommendation, "Multipoint Communication Service" [6] [27], implemented in Java with the option to use reliable multicasting communication protocols (ex: LRMP – Light-Weight Reliable Multicast Protocol [28]).

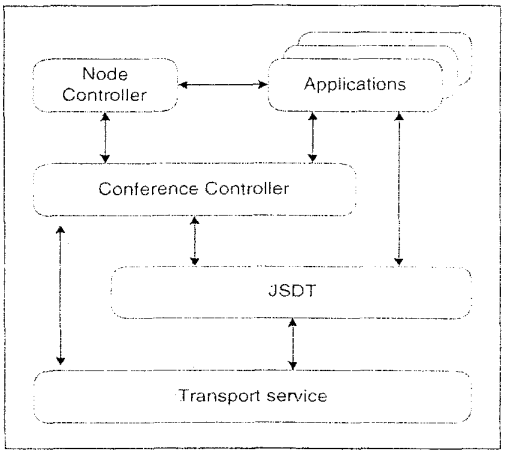

Fig. 1 – Component stack of the proposed conference system

The conference system relies on a hierarchy of nodes, clients/customers and servers, organised in a tree topology, supported by JSDT links (Figure 2). The servers are the suppliers of the conference services, with a top server and some proxy servers. The collaborative applications are in the customer nodes, linked to one or more servers, or in a server.

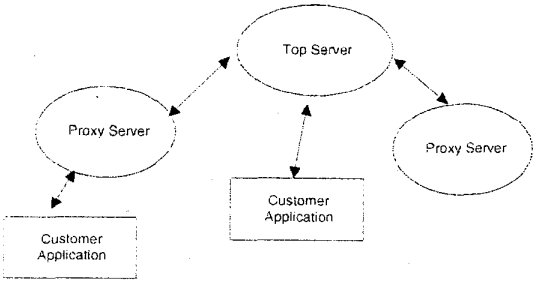

Fig. 2 – Conference typical topology

The top server manages the conference resources – sessions, channels and tokens, among others –keeping, as such, a conference information base. The resource requests are made directly to the hierarchic superior, working up in the chain until arriving at the top server. To avoid high response times, proxies have the capacity to give resources, as they keep a replication of the top server resource database. To guarantee the information updating, the top server and the proxies share a reliable multicasting communication channel. This functionality is provided by the CC module, based on the standard OMG specification CORBA *Event Service* [2] (Figure 3).

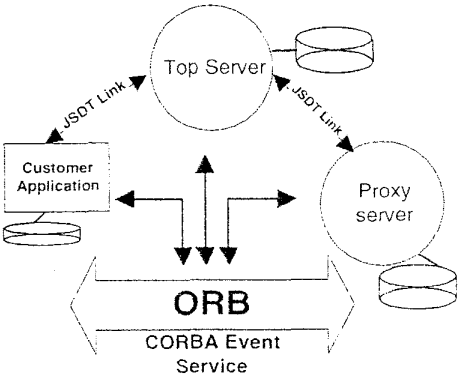

Fig. 3 – Information updating using the underlying CORBA Event Service

The CC module provides high resource availability by the use of group abstractions that comprise the top server and the proxy servers. This functionality is supported by replication processes that are based on reliable multicast communication mechanisms.

As the conference scalability is directly related to resource request response times, the servers must speed up the resource response process by placing the answers in a reliable multicasting communication channel with capacity for message filtering.

Due to its nature, proxies can assume the conference leadership, in case the top server becomes off-line for any reason. This functionality is achieved through the use of a keep-alive mechanism that is active between the top server and its proxies.

Load balancing between top server and proxies can also be implemented in systems that use the proposed architecture, as a way to improve system performance.

Conferences also require a control information database, holding information on their members and applications. This database is distributed by all nodes and keeps track of all the occurrences in each conference, as for example the entrance of a new customer node or the start-up of applications in a given node. All the changes in each node are communicated to the direct hierarchic

superior, and they will progress up the server tree until they arrive at the top server. When this happens, control information will be updated in the top server and it will be replicated back (in its totality or partially-*delta*) to all nodes down the hierarchy through the CC channels (CORBA *Event Service*).

In each node, the information base administration is performed locally by the CC, which has the advantage of having local information accesses and not accesses to the direct hierarchic superior.

The applications data are transferred directly by JSDT through public and private channels, with no need to effect their routing through the conference structure. Channels are managed by the server to which each participating node is connected. JSDT can use several transport protocols, namely reliable multicasting protocols such as LRMP – Light-Weight Reliable Multicast Protocol.

3.2. The CC reliable multicasting communication service

The CC service was developed in Java, with the reliable multicasting communication component implemented according to the CORBA Event Service model [2] (Figure 4).

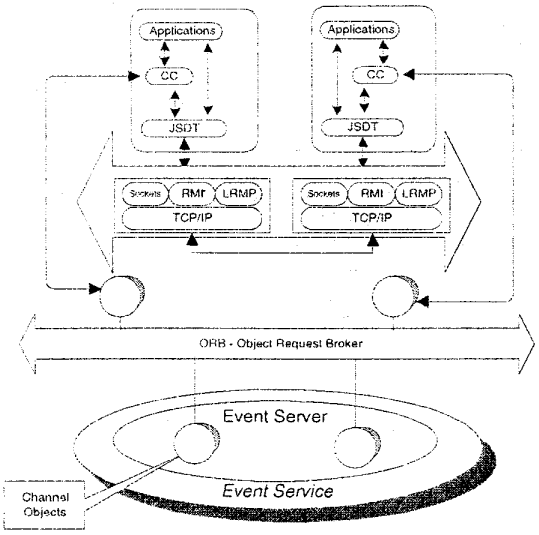

Fig. 4 – Conference system communication architecture

The CORBA *Event Service* simplifies application development because it allows asynchronous communication (event messages) between consumers and suppliers, without the need to know in advance the communicating parties, which makes it very attractive for group distributed communication [23][29].

However, this service has some limitations that span the following important issues: multicast communication environments [17][21][30], communication reliability [21][30], event filtering and correlation [29], event ordering and prioritisation [21], and "bulk" data handling [31]. In what concerns this last case, CORBA Event Service was conceived for the diffusion of small messages, so it has many limitations for non-real-time bulk multicast communication (e.g. data replication).

Our CC service was developed having in mind the need to overcome some of the above mentioned limitations. For this, some additional features were introduced, namely mapping mechanisms for native IP Multicasting (for a more efficient multicast communication), reliability process optimisation, filtering and total ordering of events/messages and fragmenting/reassembling of large messages. Moreover, as the CORBA Event Service is based on a centralised architecture, a channel is not more than another CORBA object that introduces a single point of failure [31]. With the implementation of multicast communication using native IP multicasting, channels are mapped for a specific IP multicasting address, which does not introduce a single point of failure [31], contributing for the increase of system availability.

With these refinements, we intend to provide the developed conferencing system with high scalability and performance, maintaining full compatibility with ITU T.120 recommendations.

3.3. Synchronisation

Data synchronisation is carried out by the JSDT service. This service uses a synchronisation mechanism based on a client/server architecture, i.e. all the data sent for a channel finishes being sent to the server, which redistributes them by all of the channel consumers. As the server side implementation is "single threaded" and the channels are ordered, this guarantees that the consumers receive the messages in their respective order but not at the same time. In the case of unordered channels, synchronisation can be made by a "token", which is identical to functionality of the T.120 model. In this case, the management of the "token" is made by one central object located at a server.

3.4. Implementation

Figure 5 shows the considered object model and its relationships, in Unified Modelling Language (UML).

The conference structure is initiated by the Node object. This object has the responsibility for creating the *Generic Controller* and the *ChatApplication* and the *WhiteBoardApplication*. The *Generic Controller* is the

core of the model, having the tasks of conference management, and creation and management of all of the

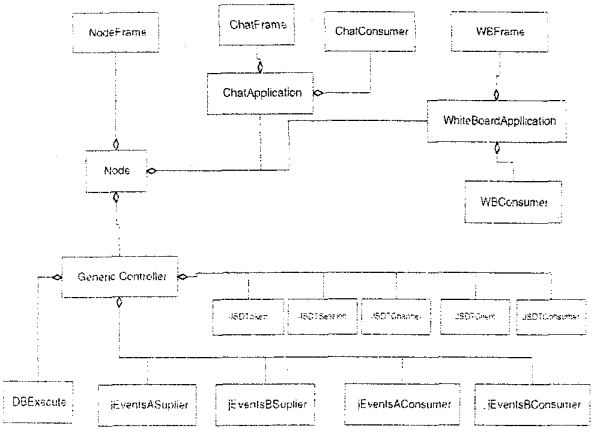

Fig. 5 – Object model of the conference service

remaining objects that constitute the physical resources and support the data communication mechanisms. When a conference is initiated by a Top Provider the *Generic Controller* interacts with the *jEventsASuplier*, *jEventsBSuplier*, *jEventsAConsumer* and *jEventsBConsumer* objects. These objects are defined using Java-IDL mapping. In addition to communicating via ORB, the conference system uses another type of communication that is based on JSDT objects. In this way, the *Generic Controller* also has the task to create a session (*JSDTSession*) and a channel (*JSDTChannel*) for the control of information exchange. When it is intended to have a private channel or when synchronisation is to be carried out in disordered channels, the system creates a *JSDToken* object. To be able to consume the available information in the channels, it is necessary to have one object that acts as a client (*JSDTClient*), which will be joined to *JSDTSession* object, and thus will be in position to "hang" one *JSDTConsumer* object for each channel that will be interested in "listening". All the interactions with databases are executed by the *DBExecute* object.

A conference prototype was developed to operate with two different types of Java collaborative applications: WB (WhiteBoard) and Chat (Figure 6). In this CC development we used the jEVENTS™ product [32], that is a Java implementation of the CORBA *Event Service* specification, without the functionalities previously specified for the CC.

4. Conclusions and guidelines for further work

In the present paper we introduced the main features of our conference control service based on the CORBA Event Service, and also a description of its specification, conception, and related prototype work. The underlying system architecture introduces some innovative mechanisms, as is the case of reliability features and the mapping of IP multicasting. Other features such as filtering mechanisms, total ordering of events/messages, and fragmentation/reassembling of large messages were also introduced. These mechanisms are at the basis of the improvement of the conference system scalability and performance. Another key aspect of the proposed system is its database replication/delta-updating process. This feature is considerably improved in the present proposal, which is due to the use of reliable multicasting communication channels. In addition, the use of group abstractions and of IP multicasting grants a high-availability to the system.

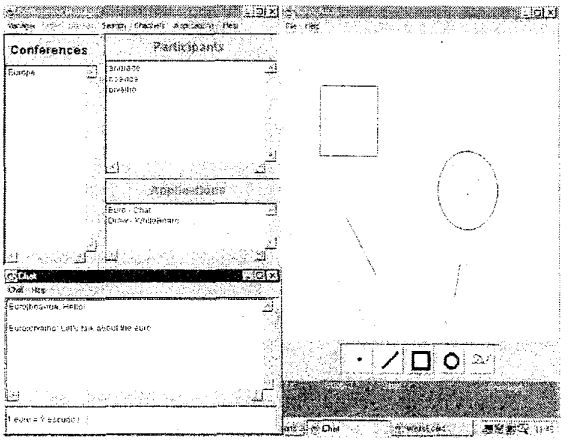

Fig. 6 – View of the prototype conference system

Thus, the proposed conference control service based on the CORBA Event Service is characterised by the following main contributions:

(1) scalability refinement of tightly-coupled conference models based on the ITU T.120 recommendations;

(2) extension of CORBA *Event Service* with reliable multicasting communication capabilities (mapping of native IP multicasting, reliability process optimisation, filtering and total ordering, and fragmentation/reassembling), adjusting it to environments where data replication/updating actions are involve, as it is the case of conference control information according to ITU T.120 model;

Our following steps will be the fine tuning and evaluation of the proposed CC CORBA Event Service. The evaluation of the implementation will be very important for the future service integration in Web environments. In addition, we plan to investigate and

explore streams treatment mechanisms [33][34] for both audio and video, and for environments with "bulk" data replication. The implementation of T.122/125 recommendations in Java (API and classes) and the exploration of the group abstraction for resource servers, as well as the respective performance studies, will be addressed in subsequent phases of the work.

5. References

[1] JavaSoft, www.javasoft.com

[2] Object Management Group-OMG (1997): "CORBAservices: Common Object Services Specification", formal/97-07-04, July1997.

[3] M. Handley, J. Crowcroft, C. Borman, J. Ott (1997): "The Internet Multimedia Conferencing Architecture", Internet-Draft, July 1997.

[4] ITU-T Study Group 8, ITU-T Recommendation T.120 (Draft, March 1995): Data Protocols for Multimedia Conferencing.

[5] ITU-T Study Group 8, ITU-T Recommendation T.121 (Draft, March 1995): Generic Application Template.

[6] ITU-T Study Group 8, ITU-T Recommendation T.122 (1995): Multipoint Communication Service for Audio Graphics and Audiovisual Conferencing.

[7] ITU-T Study Group 8, ITU-T Recommendation T.123 (Draft, June 1994): Protocol Stack for Audiographic and Audiovisual Teleconference Applications.

[8] ITU-T Study Group 8, ITU-T Recommendation T.124 (Draft, March 1995): Generic Conference Control for Audiographic and Audiovisual Teleconference Applications.

[9] ITU-T Study Group 8, ITU-T Recommendation T.124 (revised, January 1998).

[10] ITU-T Study Group 8, ITU-T Recommendation T.125 (Draft, 1995): Multipoint Communication Service Protocol Specification .

[11] ITU-T Study Group 8, ITU-T Recommendation T.126 (Draft, 1995): Still Image Transfer Protocol Specification.

[12] ITU-T Study Group 8, ITU-T Recommendation T.127: Multipoint Binary File Transfer Protocol.

[13] ITU-T Recommendation H.323 "Visual Telephone Systems and Equipment for LAN which Provide a Non-guaranteed Quality of Service", November, 1996.

[14] C. Borma, J. Ott and C. Reichert, (Work in Progress): "Simple Conference Control Protocol", Internet-Draft draft-ietf-mmusic-sccp-0x.txt.

[15] T. Helbig, S. Tretter, D. Trossen (1997): Combining CORBA and ITU-T.120 to na Efficient Conferencing Service. Proceedings of the International Workshop on Interactive Multimedia Systems and Telecommunication Services (IDMS'97), September 1997, Darmstad, Germany.

[16] D. Trossen, T. Helbig, (1997): "The ITU T.120 Series of Standards as Basics for Conference Applications", Proceedings SPIE Voice, Video and Data Communications (SPIE), November 1997, Dallas, USA.

[17] D. Trossen, K.H. Scharer (1998): "CCS: CORBA-based Conferencing Service", Proceedings of the International Workshop on Interactive Multimedia Systems ans Telecommunication Services (IDMS'98).

[18] H. Schulzrinne (1996): "Simple Conference Invitation Protocol, ftp://ftp.ietf.org/internet-drafts/draft-ietf-mmusic-scip-00.txt

[19] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson (1996): "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[20] OrbixTalk da Iona, www.iona.com/

[21] S. Maffeis, D. C. Schmidt (1997): "Constructing Reliable Distributed Communication Systems with CORBA", on topic issue Distributed Object Computing in the IEEE Communications Magazine, Vol. 14, No. 2, February 1997.

[22] R. Renesse, K. P. Birman, S. Maffeis (1996), Horus: A Flexible Group Communication System, Communication of the ACM, April 1996.

[23] S. Maffeis (1995): Adding Group Communication and Fault-Tolerance to CORBA, Proceedings of USEUNIX Conf. On Object Oriented Technologies, Monterey CA, June 1995.

[24] IMTC (1996): IMTC GCC API, ftp://ftp.imtc-files.org/imtc-site/AP-AG/TECH/CD/GCC-API

[25] IMTC (1996): Implementor's Guide, ftp://ftp.imtc-files.org/imtc-site/IMPGUIDE/impguid4.zip

[26] Java Shared Data Toolkit (JSDT), SUN Microsystems, JavaSoft Division, http://java.sun.com/people/richb/jsdt

[27] IMTC (1996): IMTC MCS API, ftp://ftp.imtc-files.org/imtc-site/AP-AG/TECH/CD/MCS-API

[28] T. Liao (1997): "WebCanal: a Multicast Web Application", Proceedings of Sixth International World Wide Web Conference, 1997, ftp://ftp.inria.fr/INRIA/Actions/webcal

[29] T. H. Harrison, D. L. Levine, Douglas C. Schmidt (1997): "The Design and Performance of a Real-time CORBA Object Event Service", Proceedings of the OOPSLA'97 conference, Atlanta, Georgia, October 1997.

[30] P. Felber, R. Guerraoui, A. Schiper (1997): "The CORBA Object Group Service", EPFL, Computer Science Department, Technical Report, 1997.

[31] K. Maad (1996): "Efficient Bulk Transfers over CORBA", Department of Computer Systems, Uppsala University, Sweden, 1996.

[32] OUTBACK, Resource Group, Inc. (1998): "jEvents Java-based Event Service", www.outbackinc.com/products/jevents/index.html

[33] Object Management Group-OMG (1997): "Control and Management of A/V Streams", specification, OMG Document telecom/97-05-07 ed., October 1997.

[34] S. Mungee, N. Surendran, Douglas C. Schmidt (1999): " The Design and Performance of a CORBA Audio/Video Streaming Service", Proceedings Hawaii International Conference on Systems Sciences, minitrack DBMS and the WWW (HICSS-32), January 1999.