

# Augmented Reliable Multicast CORBA Event Service – A Multimedia Middleware

João Orvalho<sup>\*#</sup> and Fernando Boavida<sup>#</sup>

\* - College of Education, Polytechnic Institute of Coimbra

Praca Herois do Ultramar, 3030 COIMBRA - PORTUGAL

Telf.: +351-239-793120, Fax: +351-239-401461, email: [orvalho@esec.pt](mailto:orvalho@esec.pt)

# - CISUC – Centre for Informatics and Systems of the University of Coimbra  
Communications and Telematic Services Group

Polo II, 3030 COIMBRA – PORTUGAL

Tel.: +351-239-790000, Fax: +351-239-701266, E-mail: {[orvalho](mailto:orvalho@dei.uc.pt), [boavida@dei.uc.pt](mailto:boavida@dei.uc.pt)}

## ABSTRACT

Continuous distributed interactive media applications require QoS-capable middleware platforms, in order to meet stringent requirements in terms of losses, bandwidth and jitter. Existing middleware platforms, such as OMG Common Object Request Broker Architecture (CORBA) do not meet these requirements. The major part of this paper is dedicated to the proposal, presentation and discussion of a middleware platform – named Augmented Reliable Multicast CORBA Event Service, ARMS – developed with the aim to provide quality of service mechanisms for continuous distributed interactive medium support.

**Keywords:** QoS Middleware, Reliable Multicast, CORBA Event Service

## 1. INTRODUCTION

A continuous distributed interactive medium is a medium that can change its state in response to user operations as well to the passage of time [1]. A multimedia system can be characterized by ‘computer-controlled integrated production, manipulation, presentation, storage and communication of continuous (time-dependent) or discrete (time-independent) information’ [2]. A broad variety of applications use this kind of media, such as multi-user virtual reality (VR), distributed simulations, networked games and computer-supported co-operative work (CSCW) applications. Systems dealing with multimedia events combine aspects of distributed real-time computing with the need for low latency, reduced jitter, high throughput, multi-sender/multi-receiver communications over wide area systems and different levels of reliability [3]. These event-driven systems also require efficient and scalable communications components.

For example, VR applications have specification features concerning interactivity, reliability, continuity, coherence and strict time constraints that render difficult their traffic characterisation in terms of quality of service (QoS) requirements.

In continuous distributed interactive media there are operations that need to be executed at a specific point in time and in a correct order for consistency to be achieved. [1] Investigates this problem and proposes a solution by deliberately increasing the response time in order to decrease the number of short-term inconsistencies, leading to the concept of local lag. Instead of immediately executing an operation issued by a local user, the operation is delayed for a certain amount of time before it is executed. The determination of this value is a typical issue of application adaptability.

VR applications are associated with three important functions: scalability, interaction and consistency [3]. The QoS characteristics that influence the previous functions are reliability, losses and delay jitter. Additionally, those functions are influenced by application adaptability factors like frequency of events, synchronisation delay, number of participants, consistency and playout time (display frequency) [3].

Middleware has emerged as a key architectural component in supporting distributed applications. There is a general trend towards integrating applications using reusable components based on open distributed object computing (DOC) middleware, such as CORBA [4].

Although DOC middleware is well suited for handling request-response interactions in client/server applications, the stringent QoS requirements of multimedia applications have historically precluded DOC middleware from being used as their data transfer mechanism [5]. For instance, inefficient CORBA Internet Inter-ORB Protocol (IIOP) [6] implementations perform excessive data copying and memory allocation *per-request*, which increases packet latency [7]. Likewise, inefficient marshaling/demmarshaling in DOC middleware decreases streaming data throughput [8]. Thus, in CORBA there is no support for multimedia in terms of continuous media interaction or QoS management, and it only supports request/reply interactivity, which is inappropriate for continuous media [9]. In addition, it is not possible to specify QoS for such

interactions, namely in terms of bounded latency. Also, there is no support for adaptation, as the underlying implementation is completely closed [9].

Portability over heterogeneous environments is a typical requirement of distributed multimedia systems. Although this portability can be provided by middleware such as CORBA, there exists a widespread belief in the virtual reality community that the quality of service offered by CORBA is not suitable for next-generation large distributed interactive media [10]. However, due to their nature, the CORBA Event Service [11] and the CORBA Notification Service [12] have some potential to being used for continuous distributed interactive media applications.

The work being described in this paper consists of a middleware platform for continuous distributed interactive media applications based on some extensions to the CORBA Event Service. The platform - named Augmented Reliable Multicast CORBA Event Service (ARMS), which is an intersection platform from middleware, multicast and QoS (Figure 1), that provides an end-to-end communication framework with QoS-matching capabilities, having in mind VR application requirements.

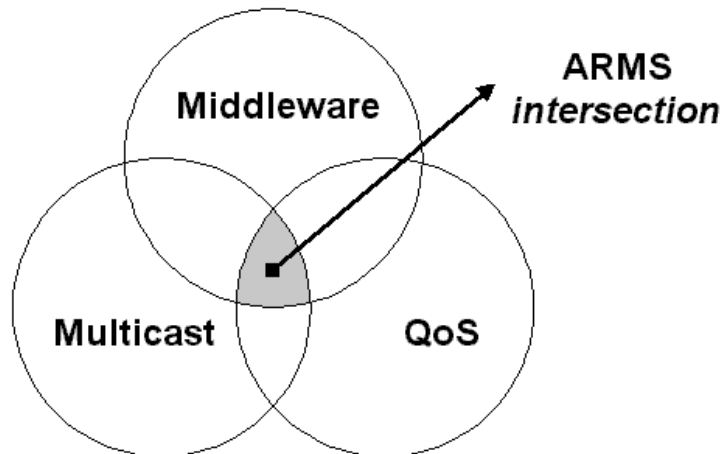


Figure 1 - ARMS *intersection*: Middleware, Multicast and QoS

The ARMS platform addresses the above-mentioned QoS issues and explores the adaptation between the quality of service required by applications and the one that is provided by the underlying communication system. This paper presents the main approaches taken by ARMS in order to provide QoS adaptability. Section 2 provides a general description of the ARMS architecture. As the work presented in this paper corresponds to an enhancement of a previous, non-QoS-adaptive platform, section 3 presents the characteristics of this previous platform. Section 4 provides details concerning the approach taken by ARMS in terms of reliability, congestion control and jitter. Section 5 describes additional features of the ARMS platform, namely the IIOP/IP multicasting gateway service and the federation of event channels. Section 6 identifies related work. The conclusions are presented in section 7.

## 2. ARMS GENERAL CHARACTERISTICS

ARMS is focused on VR environments based on VRML models, which deal with two different types of information: events and states. Events are time critical and described by small amounts of information, as opposed to states that are generally non-time-critical and require large amounts of information for their description. Therefore, there is the need for different levels of reliability when exchanging event and state information: minimal reliability (with loss detection) for events, and full reliability for states. In addition, ARMS assumes that there is a common time reference, which requires the clocks of all participants to be synchronised by means of NTP or GPS clocks. Previous research has been focused on limitations of the CORBA Event Service, namely multicasting, reliability and bulk data handling [13,14]. The work presented in this paper extends it to support adaptive QoS middleware functionalities. ARMS offers a set of QoS-related mechanisms for reliability guarantee, congestion control and jitter control. The QoS management process is supported by object-based monitoring and adaptation functions (Figure 2). Monitoring is the process of observing the utilisation of resources and/or QoS characteristics in the system. ARMS has specific objects for loss and jitter monitoring.

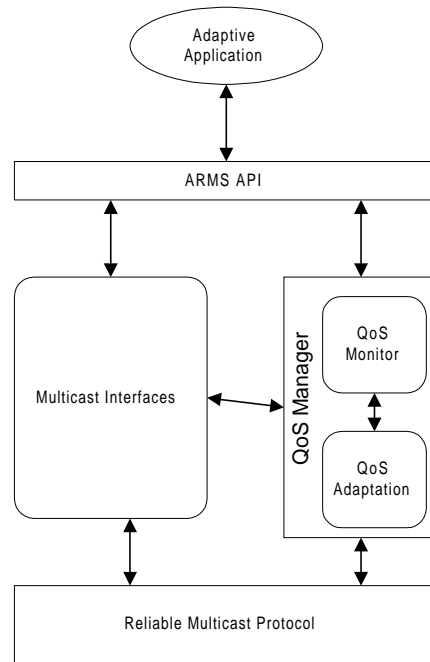


Figure 2 - ARMS architecture

Adaptation mechanisms generally rely on resource control, reconfiguration or change of service [15]. ARMS uses a resource control paradigm, providing adaptation mechanisms for network congestion, losses and jitter. Placing adaptation capabilities in the middleware gives applications the ability to concentrate on specific functionalities, to enforce different adaptation policies and to interact with other components in the system in order to ensure fairness and other global properties [16].

### 3. ARMS VERSION 1

The first generation of ARMS offered a set of extensions to the CORBA Event Service specification [13], namely mechanisms for IP multicast communication, reliability and fragmenting/reassembling of large events.

This service is a pure Java implementation (JDK 1.2 compatible) [17], and so it benefits from all the strengths of Java. These include portability, security, and robustness. It is, also, ORB-agnostic. It is written to the standard IDL-to-Java mapping, so it should work with any Java ORB that supports the standard mapping. As in the original specifications made by OMG CORBA Event Service, suppliers and consumers are decoupled, that is, they don't know each other's identities; thus, the standard Event Service may still be provided by this extended service.

The approach on which the developed work was based is the one called *Push* model. The objective is to allow the consumer to be sent the event data as soon as it is produced. The canonical Push Model allows Suppliers of events to initiate the transfer of event data to Consumers.

The reliable multicast extension can be seen as an alternative way to get the event across to the consumer. This assumption forces the new service to keep all the standard interfaces with the same functionality (the same methods) that are defined by OMG, allowing a choice to be made by the supplier/consumer between IIOP and Reliable IP Multicast. The service implements two kinds of IP multicast interfaces: IP Multicast-Any and IP Multicast-Streams. The IP Multicast-Any deals with *Any* Values while IP Multicast-Streams deals directly with byte-stream values, avoiding the overhead caused by marshalling and de-marshalling of the proprietary *Any*. The reliable multicast solution is based on the Light-weight Reliable Multicast Protocol (LRMP) [18], which deals with IP Multicasting and provides the necessary reliability and better scalability.

### 4. ARMS VERSION 2

The quality of service provided to multimedia sessions is determined, in general, by packet losses and delay. The second generation of ARMS acts as a transport level with QoS monitors that provide information on key aspects of QoS provision such as losses and jitter. Adaptation mechanisms for network congestion (congestion control), losses (reliability sessions) and jitter (jitter filter) have an important influence on the performance of virtual environment applications.

#### 4.1 Reliability sessions

ARMSv2 supports several reliability levels: *Reliable*, *Reliable-Limited Loss*, *Loss Allowed*, *Unreliable with loss notification* and *Unreliable*.

The first is a typical strong reliability session. *Reliable-Limited Loss* is a session where limited losses are permitted for some types of packets and loss notifications will be triggered when this happens. It provides guarantees for congestion control at sender side and sequence control at the receiver side. *Loss Allowed* is a session where losses are allowed and accepted for all types of packets but provides loss notification, congestion control at sender side and sequence control at the receiver side. *Unreliable with loss notification* is an unreliable session, where data are not subject to congestion control at the sender side. However, the ARMS upper level maintains a queue for sequence number control, which allows loss notification. Lastly, *Unreliable* is a purely unreliable session. Table 1 summarises the characteristics of the various types of sessions offered by ARMSv2.

Table 1 – Types of ARMSv2 sessions

	Permitted Losses	Loss Notification	Congestion Control	Sequence Control
<i>Reliable</i>	No	Yes	Yes	Yes
<i>Reliable-Limited Loss</i>	Yes, for some data types	Yes	Yes	Yes
<i>Loss Allowed</i>	Yes	Yes	Yes	Yes
<i>Unreliable with loss notification</i>	Yes	Yes	No	Yes
<i>Unreliable</i>	Yes	No	No	No

#### 4.2 Congestion control

Currently, different approaches are being discussed to solve QoS and congestion control problems [19]. Complex distributed applications residing in heterogeneous end-to-end computing environments must be flexible and adapt to QoS variations in their end-to-end execution [16]. That is, applications must adapt the bandwidth share they are using to the network congestion state [19]. Usually, there are two distinct levels at which adaptation may take place – the system level (e.g. operating systems and network protocols) and the application level – with different objectives. In order to balance the objectives of these approaches, the ARMS middleware closely interacts both with application needs and with multicast protocols, monitoring network parameters and operating systems resources. In terms of network QoS control mechanisms, ARMS directly monitors the reliable multicast communication protocol, LRMP, adapting the sender transmission rate to the network congestion state. The adaptation is based on information carried by NACK packets and on local congestion information [14] as the sender window size [18]. Based on congestion information gathered from lower communication objects, ARMS and applications adjust the upper-level sending rate. QoS mechanisms that are based on adapting the sender transmission rate to the network congestion state don't work well in large multicast groups and heterogeneous environments, because poor performance receivers would impose a low transmission quality. To avoid this, several proposals have been made for hierarchical data distribution [20,21,22]. Nevertheless, in virtual reality environments, data layering approaches are not appropriated for most data, especially for time critical data such as VRML events. Delay is the most important QoS factor for this type of data. Layered data mechanisms solve heterogeneity problems but cause additional delay at the receivers [19]. Thus, to avoid this, ARMS adapts the minimum rate to ensure the fairness of the adaptive congestion control. Receivers should leave the session when the loss rate is very high and the data rate is not reduced by the sender. Nevertheless, layered multicast [20] can be useful and will be explored in subsequent stages of the work.

#### 4.3 Jitter

Variance in end-to-end delay is called delay jitter or, simply, jitter. Critical information such as the case of audio, video and continuous distributed interactive media (distributed simulations, multi-user virtual reality) should be played back continuously, which means that there must be some form of jitter compensation.

In addition to components for error control, ARMS provides mechanisms for monitoring and controlling jitter, so that the original temporal relationships can be recovered. ARMS lower level reliable multicast protocol queue sends ordered packets (reliable data) to an upper level Jitter Filter Queue. This Jitter Filter Queue is used to absorb delay variations exhibited by arriving packets. So, this compensation is done by introducing an additional and variable delay, followed by the delivering of packets to the application level, as opposed to a compensation made at the application level [23].

ARMS allows applications to contract this jitter compensation [the threshold synchronisation] for certain types of data, namely unreliable data (pure or with loss notification) and reliable data with limited loss. Packets that arrive after a given threshold are considered to be too late. These can simply be dropped or, alternatively, be marked as late packets and passed to the application. Applications can ignore these packets or can react by requesting that a special NACK be sent by ARMS.

## 5. ADDITIONAL FEATURES OF THE ARMS PLATFORM

In addition to providing QoS adaptation, the ARMS platform has some optimisation features that contribute to its transparency and scalability. With regard to transparency, the platform provides a gateway service between IIOP and IP Multicasting. Scalability is supported by the federation of Event Channels.

### 5.1 Event Channel IIOP/IP Multicasting Gateway Service

ARMS guarantees interoperability between event suppliers and consumers in a way that is independent of the used communication facilities – standard IIOP or IP multicast. To achieve this, the Event Channel provides a transparent IIOP gateway. As can easily be understood, the interfaces that are provided by OMG must be maintained by the proposed service in order to deal with the *Any* type of data. In this way, interoperability between the proposed service and most of the commercially available ORBs can be achieved. In order to deal with all the values supported by the *Any* type, a new object – *Any* – was created as an extension of the OMG class, to override the ORBs implementation.

The values to be passed to this service, in order to be sent via multicast, shall be created with the proprietary interfaces of the service, that provide a set of new methods to deal with the resulting buffers and fill the LRMP data packet with the data contained in those buffers. So, by creating a method for extracting the sequence of bytes generated by the *Any* marshalling operation, the result is a stream of bytes that mean nothing to LRMP. Once they get across to the consumer LRMP object, it reassembles them to form an *Any*, and delivers them to the consumer as a valid *Any* that shall be extracted by the application interfaces. To provide the interchange of Multicast events and Standard events, two scenarios are considered (Figure 3):

- when the supplier is a *MulticastPushSupplier* and the consumer is a normal Consumer;
- when the supplier is a normal Supplier and consumer is a *MulticastPushConsumer*.

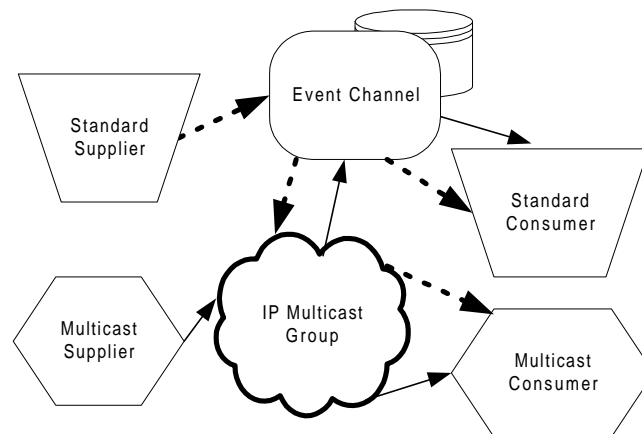


Figure 3 – ARMS Gateway service data flow

For the first scenario, and knowing that the model to be dealt with is the Push Model, the event is sent via LRMP, i.e. is sent to a well known multicast group, and is pushed to Multicast Consumers by one LRMP object that receives events directly. As the Event Channel holds one *ProxyMulticastPushConsumer* for all Suppliers, this proxy acts as a normal consumer to which events are pushed. Each time this proxy is pushed a new event, it invokes the receive method on the Event Channel. This invokes a receive method on the *ConsumerAdmin*, that holds references to all *ProxyPushSuppliers* (standard) and shall invoke the receive method on all those proxies, that will then invoke the push method on the consumer they are attached to. The *ConsumerAdmin* does nothing on the *ProxyMulticastPushSupplier*, or else events would be sent twice to the multicast group.

For the second scenario, the supplier calls the push method on its *ProxyPushConsumer*, that will call receive on the Event Channel, to get the event to standard consumers, and will also ask for any existent *ProxyMulticastPushSuppliers*. If there are any, the proxy shall call the receive method on that proxy. In this way, the event is sent to the multicast group, and all *MulticastPushConsumers* receive it. So, the Event Channel is responsible for doing all the necessary IIOP gateway work.

In either scenario, the gateway operation implies changes to the Any Object that is being forwarded. The Event Channel also takes care of this detail, at the cost of an additional overhead caused by the de-marshalling of the proprietary Any and subsequent marshalling as an ORB Any, and vice-versa.

### 5.2 Federation of Event Channels

Distributed transparency of the Event Channel can lead to a less effective configuration. There are scenarios where consumers and suppliers reside in the same process, host or network and the Event Channel is remote. In these cases, there is a waste of network resources and unnecessary increase of latency. ARMS Event Channel object uses a configuration facility to federate Event Channels, allowing an Event Channel to be a consumer of another, remote Event Channel. Federation of Event Channels leads to the conservation of bandwidth because only a single event will be sent to all the remote users. Additionally, the average latency is reduced because part of the traffic becomes local. Figure 4 illustrates the use of the federation of event channels when communication is made via standard IIOP. In Figure 5 federation of event channels is used in conjunction with the IIOP/IP Multicast Gateway Service. The combination of IIOP/IP Multicasting Gateway Service with Event Channel federation contributes to the enhancement of the QoS characteristics of the platform, namely in terms of transparency, latency and scalability.

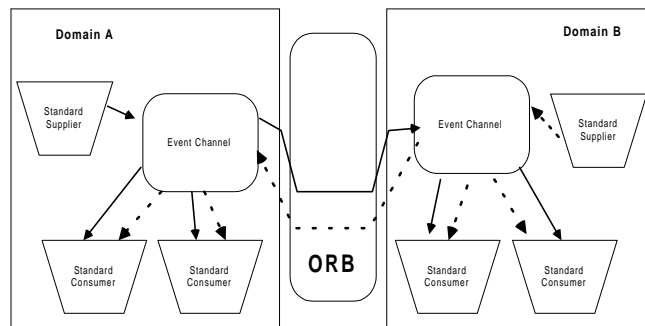


Figure 4 – Federated Event Channel configuration: standard IIOP

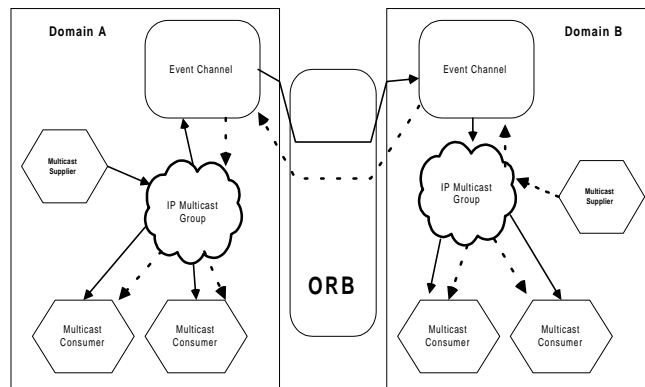


Figure 5 – Federated Event Channel configuration with IIOP/IP Multicasting Gateway Service

## 6. RELATED WORK

Several approaches exist that try to explore QoS issues in distributed object environments. OrbixTalk [24,25], which is a commercial IONA implementation of the CORBA *Event Service* specification normalised by OMG [11] written in C++, uses IP multicasting and a reliability mechanism based on *negative acknowledgements*, in order to provide delivery confirmation of each information object. Because the CORBA *Event Service* specification does not address issues for real-time applications, the QoS behaviour is not acceptable for many application domains.

OMG Notification Service [12] is a superset of the CORBA *Event Service* specification. It adds some interfaces, which deal with filtering, security, federation and QoS. However, this specification does not address implementation issues.

TAO's Real Time Events Service [10] it's a powerful implementation of CORBA *Event Service and Notification Service* specifications in C++. Nevertheless, it lacks some QoS characteristics, such as reliability. However, OMG CORBA Messaging specification [26] defines several levels of reliability for one-way calls, which are being added to the TAO

features to complement this service. This group has been developing considerable work on real-time extensions to CORBA, which enable end-to-end QoS specification and enforcement.

Other CORBA projects, such as QuO [27], implement models for distributed object application, defining, controlling, monitoring and adapting to changes in QoS parameters. The QuO project proposes various extensions to standard CORBA components and services, in order to support adaptation, delegation and renegotiation services to shield QoS variations. The development has a great focus on remote method invocation to remote objects.

ARMS is complementary to the above-mentioned approaches, since it is based on the integration of CORBA middleware and underlying services – such as multicasting – while the referred approaches are concentrated on the CORBA object model.

## 7. CONCLUSIONS

Quality of service requirements of continuous distributed interactive media applications encompasses several aspects that are not readily available in standard distributed object platforms. The need for different levels of reliability, congestion control mechanisms and jitter suppression strategies is apparent in applications such as virtual reality, CSCW and distributed interactive simulations.

Middleware platforms can play an important role in quality of service provision, offering flexible mechanisms that adapt the quality of service provided by the underlying communication channel to the quality of service required by applications, according to an established service contract.

This paper presented the main implementation options of the ARMS middleware platform, which builds on a set of extensions to the CORBA Event Service, providing native multicast communication, various reliability levels, congestion control and jitter suppression, with the aim to achieve QoS adaptability. The platform has been implemented at the Laboratory of Communications and Telematics of CISUC, where it is operational and currently being subject to extensive evaluation.

The first set of tests made to the platform was aimed at the verification of the platform operational status. The basic mechanisms for reliability, congestion control and jitter proved to be operational. The following tests will try to quantify the usefulness and effectiveness of these mechanisms. These will provide valuable information concerning the adequacy of placing QoS adaptation mechanisms in the middleware, as opposed to strategies that place them at application level.

## 8. ACKNOWLEDGEMENT

This work was partially financed by the Portuguese Foundation for Science and Technology, FCT and PRODEP.

## 9. REFERENCES

1. "Consistency in Continuous Distributed Interactive media", Martin Mauve, Technical Report TR-9-99, Reihe Informatik, Department for Mathematics and Computer Science, University of Mannheim, November 1999.
2. "Human perception of jitter and media synchronization", IEEE J. Select. Areas Commun., vol. 14, pp. 61-72, January 1996.
3. "Quality of Service Management in Distributed Interactive Virtual Environment", Dimitrios Makrakis, Abdelhakim Hafid, Farid Nait-Abdesselem, Anastasios Kasiolas, Lijia Qin, Progress Report of DIVE project.  
[http://www.mcrlab.uottawa.ca/research/QoS\\_DIVE\\_Report.html](http://www.mcrlab.uottawa.ca/research/QoS_DIVE_Report.html)
4. Object Management Group, *The Common Object Request Broker: Architecture and Specification*. Object, Management Group, 2.3 edition. 1999
5. Design and Performance of an Object-Oriented Framework for High-Performance Electronic Medical Imaging. Pyarali, I., Harrison, T. H., and Schmidt, D. C. , *USENIX Computing Systems*, 9(4). 1996
6. Optimizing a CORBA IIOP Protocol Engine for Minimal Footprint Multimedia Systems. Gokhale, A. and Schmidt, D. C. *Journal on Selected Areas in Communications special issue on Service Enabling Platforms for Networked Multimedia Systems*, 17(9). 1999
7. Measuring and Optimizing CORBA Latency and Scalability Over High-speed Networks, Gokhale, A. and Schmidt, D. C., *Transactions on Computing*, 47(4). 1998
8. Measuring the Performance of Communication Middleware on High-Speed Networks, Gokhale, A. and Schmidt, D. C., In *Proceedings of SIGCOMM '96*, pages 306–317, Stanford, CA. ACM. 1996
9. "Supporting Multimedia in Distributed Object Environments", D. G. Waddington and D. Hutchinson, *Internal report number MPG-99-17*, 1999.
10. "Applying a Scalable CORBA Event Service to Large-scale Distributed Interactive Simulations", Carlos O'Ryan, David L. Levine, Douglas C. Schmidt, J. Russel Noseworthy, Proceedings of the "5Th Workshop on Object-oriented Real-time Dependable Systems", Monterey, CA, November 1999.
11. "CORBA services: Common Object Services Specification", Object Management Group-OMG, Document formal/97-07-04, ed., July 1997.
12. "Notification Service Specification", Object Management Group-OMG, Document telecom/99-07-01, ed., July 1999.
13. "A Platform for the Study of Reliable Multicasting Extensions to CORBA Event Service", João Orvalho, Luis Figueiredo, T. Andrade and Fernando Boavida, Proceedings of IDMS'99, Toulouse, France, October 12-15, 1999.
14. "Evaluating Light-weight Reliable Multicast Protocol Extensions to the CORBA Event Service", João Orvalho, Luís Figueiredo and Fernando Boavida, , in the Proceedings of EDOC'99 – 3rd International Enterprise Distributed Object Computing Conference, University of Mannheim, Germany, September 27-30, 1999.
15. "A General Model for QoS Adaptation", D. G. Waddington and D. Hutchinson, Proceedings of Sixth International Workshop on Quality of Service (IWQoS'98), pag.275-277, Napa, California, USA, May 18-20 1998.

17. "A Control-Based Middleware Framework for Quality-of-Service Adaptations", Baochun Li and Klara Nahrstedt, IEEE Journal On Selected Areas In Communications, Vol. 17, N° 9, September 1999.
18. JavaSoft, [www.javasoft.com](http://www.javasoft.com)
19. Tie Liao: "Light-weight Reliable Multicast Protocol Specification", Internet-Draft: draft-liao-lrmp-00.txt, 13 October 1998.
20. "The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme", Dorgham Sisalem and Henning Schulzrinne, In Proceedings of Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'98), Cambridge, UK, 1998.
21. "TCP-like Congestion Control for Layered Multicast Data Transfer", L. Vicisano, L. Rizzo, and J. Crowcroft, In Proceedings of INFOCOM'98, San Francisco, USA, March 1998.
22. "Receiver-driven layered multicast", S. McCanne, V. Jacobson, and M. Vertteli, In Proceedings of SIGCOMM Symposium on Communication Architecture and Implementation of High Performance Communication Systems (HPCS'97), Chalkidiki, Greece, June 1997.
23. "Thin Streams: An Architecture for multicasting layered video", L. Wu, R. Sharma, and B. Smith, In Proceedings of Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'97), St. Louis, USA, May 1997.
24. "Delay and Synchronization Control Middleware to Support Real-Time Multimedia Services over Wireless PCS Networks", H. Liu and M. El Zarki, IEEE Journal On Selected Areas In Communications, Vol. 17, N° 9, September 1999.
25. OrbixTalk da Iona, [www.iona.com/](http://www.iona.com/)
26. "Reliable CORBA Event Channels" X. Défago, P. Felber, R. Guerraoui, EPFL, Computer Science Department, Technical Report 97/229, May 1997.
27. "CORBA Messaging Specification", Object Management Group-OMG, Document orbos/98-05-05, ed., May 1998.
28. "QuO's Runtime Support for Quality of Service in Distributed Objects", Vanegas R, Zinky JA, Loyall JP, Karr DA, Schantz RE, Bakken DE.. Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98), The Lake District, England, 15-18 September 1998.