Prototyping QoS based Architecture for Power Plant Control Applications

C. Lizzi[*], L. Bacon[++], E. Becquet[*], E. Gressier-Soudan[*]

[*]CEDRIC-CNAM,
292 rue St Martin,
75 141 Paris Cedex 03 France,
{lizzi, becquet, gressier}@cnam.fr

[++]EDF, DRD/EP/CCC/Groupe Architecture,
6 Quai Watier,
78 401 Chatou Cedex, France
laurent.bacon@edf.fr

Abstract

The goal of the project in this paper is to gain experience on QoS (Quality of Service) based distributed system for EDF's power control applications. EDF is the French power utility. QoS is related to time constraints in our case. We adopted a distributed system approach to build our execution platform. The first prototype uses a modified ChorusOS micro-kernel with a real-time inter-process communication facility based on an ATM network and supports a specific real-time java virtual machine. Our second proposal attempts to build an equivalent architecture using off-the-shelf product. Reality constraints leads to a CoS (Classification of Service) based design. We show that there is no available off-the-shelf QoS nor CoS based technology today for real-time object oriented distributed applications.

1. Introduction

The goal of the project described in this paper is to gain experience on QoS (Quality of Service) based distributed system for EDF's power plant control applications. EDF is the french power utility. QoS is related to time constraints in our case, it does not deal with reliability nor security constraints.

Process control applications are naturally distributed. They respect the CIM (Computer Integrated Manufacturing) hierarchy [43] different levels: level 0 deals with captors and actuators, level 1 supports acquisition and control, level 2 performs supervision functions, level 3 deals with plant production control. Current solutions are dedicated monolithic architecture, message oriented distributed applications, built on top of OSI or proprietary protocol stacks. Generally, networks are heterogeneous: fieldbuses [44] [35] or point to point links at the lower levels (1 and 2), LANs at the other levels. Operating systems are heterogeneous too: real-time micro-kernels at level 1, general purpose reliable kernels at level 2 or higher, and then different suppliers are involved.

We adopted a distributed system approach to build our architecture. This distributed architecture is supposed to support cooperating objects as it is required by modern platforms [10], [20], [34], [16]. Interactions are subject to with real-time constraints. We selected QoS as a design paradigm to map real-time constraints from the application [5]. It offers a uniform way to handle time requirements as delay, jitter and throughput at any level of the architecture, including application level. The concept of loss ratio issued from network technologies as ATM can be retained as an application QoS parameter. It keeps its meaning with task scheduling, and can represent the ratio of eliminated tasks missing their deadline. The real problem comes up with the way QoS is implemented in the execution platform.

The first platform that we have designed addresses real-time constraints with an homogeneous environment built from scratch. The first execution platform is a prototype. It has been built in the context of a join project between our lab, CNAM-Cedric, and a company, CSTI in Lyon [25]. It uses on a modified ChorusOS micro-kernel [25] with a real-time inter-process communication facility [22] based on an ATM network [21] and offers a specific real-time java virtual machine to application entities [24]. MidArt [28] is an other example of a ATM based middleware architecture for process supervision.

From an industrial point of view, it seems to be not realistic to address all the parts of a process control application with the same system platform, especially when you consider the cost of ATM technology. This approach helped us to focus on design properties that should be met with an off-the-shelf products based platform. Therefore, our second proposal attempts to build an equivalent architecture using off-the-shelf product. Real-time constraints leads to a CoS (Classification of Service) based design [1]. Usually CoS is dedicated to DiffServ Internet networks, but this concept can be extended to execution platform and applies to distributed systems and to applications. We show that despite the interest of a QoS nor a CoS object oriented distributed system approach, one can say that

there is no available off-the-shelf products today for real-time distributed applications.

This paper is organized as follows. Section 2 presents the key features of the power plant control application. Section 3 describes a first prototype based on a modified ChorusOS micro-kernel with a real-time inter-process communication facility over an ATM network that offers a real-time java virtual machine. Section 4 presents an alternate solution built with off-the-shelf components. Section 5 concludes and presents future directions of our project.

2. Application Characteristics

The main activity of EDF is to produce energy. You can see on figure 1, a typical example of an electricity production process and typical functions we are going to supervise. Cassical thermic power plants are based on a water/steam cycle. Water, when presented to a hot source, is transformed to an optimal steam in terms of electricity production. Then this steam is injected in a turbine coupled with an alternator. This last element produces directly the power. Finally, the used hot steam needs to be freshened into water. A new cycle can be started.

To allow this cycle, there is a lot of different elements in the plant: pumps (especially water pumps which assure water transportation in the whole circuit), tanks (water, fuel), valves, transformers, circuit breakers… In fact, the process presented in figure 1 is more complex. Conducting teams deal with this complexity, they can conduct the plant manually. But more control command systems are continuously added to help them. It helps by acquiring data about plant elements (such as valves states :open or closed), by transmit commands directly to the same elements, by presenting process state in a graphical and more usable way, by doing auto-regulation and more. These actions need reliability. In most of the cases, this is achieved by redundant systems.

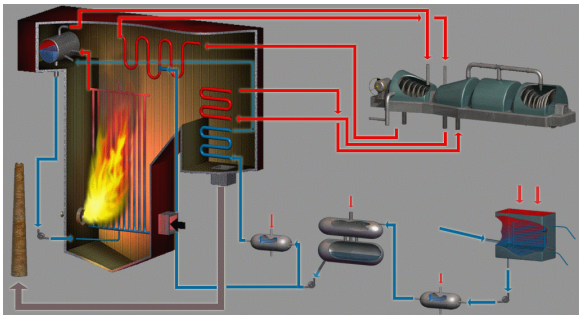

Figure 1. Thermal Power Production Process (Thermo-electricity)

The application is naturally distributed. Control devices are distributed along the process and control functions are mapped on different computers with replication schemes for fault tolerance purpose. The classical design of such applications is layered by the CIM model. Our work targets the low levels of the control application: level 0 deals with captors and actuators, level 1 supports acquisition and control, level 2 performs supervision functions (operator interface, process regulation, alarms and states recording, printing …). Current solutions are cost effective dedicated monolithic architecture, message oriented distributed applications, built on top of OSI or proprietary protocol stacks. Operating systems and hardware are fully heterogeneous. RT constraints are soft: the application end-to-end delays (from level 0 to level 2) are about 500ms, no information losses are allowed and solutions should run at least during thirty years. This architecture is described on figure 2 and 3, both hardware and software point of views.

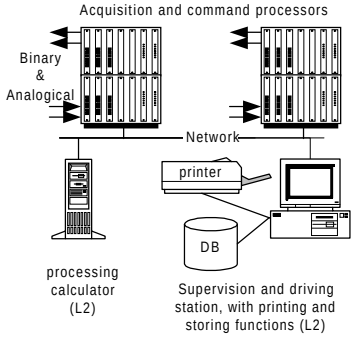

Figure 2. Representative Platform

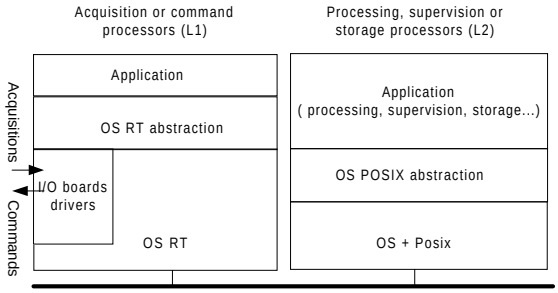

Figure 3. Representative Software Architecture

Figure 4 shows data flows between level one and level two calculators. In our next platform, we want to replace the ATM network with a fast Ethernet, and to use it for levels one, two and three.

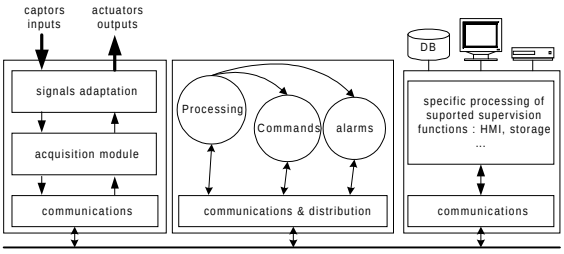

Figure 4. Control-Command data streams

3. An ATM based Real-Time Distributed System Architecture

3.1. Principles of the platform

The design of the platform is born from CSTI's requirements to address real-time applications encountered in the process control and manufacturing domains. Initial CSTI's platform, ANTARA [12], choices were : PowerPC based boards, ChorusOS as real-time micro-kernel, and a 155Mb/s ATM network for communications. The micro-kernel is supposed to drive actuators and captors directly, or programmable logic controllers via a fieldbus system. Figure 5 gives an overview of an application targetted by CSTI's distributed system. Equivalent requirements are made by the MidArt platform [28].

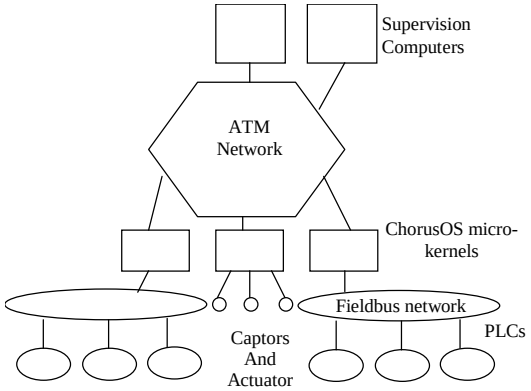

Figure 5. ANTARA platform for Process Control

To meet hard real-time constraint requirements, the ChorusOS micro-kernel has been enhanced. Enhancements are described below and deals with scheduling, priority inheritance, buffer management and distributed inter-process communications. On top of this distributed system, a java virtual machine has been ported to support distributed objects.

3.2. Extensions to the ChorusOS microkernel

ChorusOS is a micro-kernel based distributed real-time operating system. Its kernel scheduler implements a pre-emptive highest priority first (HPF) scheduling. Affectation of kernel priorities to the various tasks is performed by upper scheduling classes that implement policies. Users never manage the kernel priority of tasks, but have to provide scheduling parameters that are specific to each class. One of the problems with respect to real-time scheduling is that applications may have distinct metrics to express the criticality of their tasks. They may notably be expressed in terms of urgency or importance, or as a combination of both. The new scheduling architecture enables the implementation of deadline-based scheduling policies. This framework retains a clear separation between generic kernel mechanisms and upper level scheduling classes.

We schedule real-time tasks using the concept of generalized priority, defined as a tuple whose terms constitute a set of scheduling criteria. Generalized priorities are made of a criticality level, an absolute deadline and a logical timestamp. Criticality comes into play only during overloads. During underloads, all tasks are expected to share the same criticality, so that a pure Earliest Deadline First (EDF) scheduling is observed [40]. With overloads, criticality help to determine the subset of tasks that would meet their deadline. Tasks are additionally stamped by positive or negative values upon rescheduling events: a task that becomes runnable is executed after any other task already runnable that has the same criticality and the same deadline, a task which is preempted by another one will be later re-elected before any other task already runnable that has the same criticality and the same deadline. With this framework, a pure earliest deadline first algorithm can be implemented, but enhanced policies such as D-over [19] or Robust EDF [9] can be also supported. This framework preserves compatibility with the original priority-based scheduling classes shipped with the on-the-shelf ChorusOS, original classes can be scheduled on a per criticality basis.

The modifications have slightly lowered the overall performances of the system. For instance, when 2 (respectively 100) tasks compete for the processor with a standard micro-kernel, task switches perform in 1.1µs (respectively 4.2µs) on 300 MHz PowerPC hosts. With the new scheduling framework, this time now varies from 1.5µs to 6.4µs, depending on the number of runnable tasks in the scheduler. From an application point of view, the drawback of this overhead is "compensated" by the ability to do deadline and criticity-based scheduling.

The ChorusOS real-time mutexes support priority inheritance. They have been modified to preserve our new generalized priority-based model. More details are given in [24][25]. This framework can be used by the Java virtual machine.

3.3. Real-time communications over ATM

ATM has been chosen for its capability to handle QoS constrained streams. It emerges from users requirements [2] that power plant supervision needs audio and video information. Therefore, ATM is an attractive technology to merge control-command and audio/video streams in an homogeneous way.

3.3.1. Clock Synchronization

The ability to synchronize the clocks of a distributed system can be an important feature to provide real-time distributed services or to maintain the coherence of distributed data. We needed this synchronization because our system allows absolute deadlines propagation and inheritance. So, absolute deadlines must have a global meaning in the system.

There is two different kinds of clock synhronization in a distributed system: internal and external. External synchronization is based on an external time reference, such as TDF in France or Naval Observatory in USA. Internal synchronization is based on an internal time, time given by the clock of one of the distributed system's computers. External approach permit interactions with farest distributed systems but is a centralized approach: if your reference source or equipment used to get that time fails, there is no more time synchronization in the system. On the other hand, the internal approach is more fault tolerant but is only suited for intranet designs, isolated from external world.

We use a hybrid approach which combines internal and external solutions as described above, to achieve certain properties [15]. First, when an external source is available, clocks are synchronized with it. Second, if that absolute reference is no more available (whatever the reason is), clocks are synchronized in an internal way. Clock synchronization remains valid even if distributed system's sites or determined clocks fails. Finally, synchronization is accurate enough, even when external time reference is poor quality.

We use GPS as the external time reference. Sites periodically exchange stamps to achieve the synhronization. Each site synchronizes with the external source if it is available, in an internal way else. The use of ATM capabilities such as CBR (Constant Bit Rate) circuits, that offer bounded delays and bounded jitter, allows at the end a synchronization accuracy better than 100 µs.

3.3.2. The Real-Time IPC in the extended ChorusOS

ChorusOS is not only a real-time micro-kernel but also a distributed system. It offers a D-IPC (Distributed Inter-Processus Communication) facility that allows location independant communications between entities. From a real-time point of view, it delivers a best-effort service. It does not offer any guarantee about transmission delays and works in a non-connected mode. It means that every message is sent separately, as stand-alone data units, and that you can't do resources reservation.

Because the applications targeted need a real-time communicating system, a new D-IPC service has been defined. It includes real-time support and works in connected mode. This new D-IPC takes advantage of the underlying ATM network. As the original ChorusOS D-IPC, communications between tasks (local or distant) are identical and remains location independant.

The messages sent over this new real-time D-IPC is slightly different. First, it is allocated statically at the beginning of the task, to prevent memory allocation or swap overhead. Second, it contains an absolute deadline (defined at message initialization or inherited from the sending task). This deadline is transmitted to the receiving task and has to keep its meaning. This is the reason why clock synchronization has been introduced.

The real-time port is represented by an object with three interfaces, one for each server/client of the communication service and one for control. This model with the three interfaces makes this real-time service compliant to the RM-ODP reference model [5].

We introduce another object called the binding object. It is an object between two real-time ports involved in the communication, source port and target port. This model is compliant with the RM-ODP model and is directly linked to the concept of explicit binding [41]. A binding object has also three interfaces, two for connection with source and target real-time ports, and one interface to control the binding state (monitoring).

To have real-time communications, you have to be able to ensure properties about delays, jitter and others that permit you to bound end-to-end delays. These communication characteristics are commonly called the Quality of Service.

Connection request is subject to admission control, in order to ensure that the QoS required by the application can be provided. In the case of a local communication, this step always succeeds. Admission tests are based on a description of the connection that details the characteristics of the data traffic being conveyed and the QoS expected from the RT-IPC service. Traffic specification is expressed in term of peak and substainable rates, and maximum message size, while the QoS specification includes transfer delay and maximum admissible jitter. Traffic and QoS specifications compose the flow specification (flowspec). For any connection establishment request, two flowspecs are submitted: a target flowspec, that denotes the nominal service level for the connection being requested and an acceptable flowspec, that defines the minimum service level acceptable for the connection. If the service provider is unable to establish the connection at the requested target level, it should try to establish it at the best possible level, greater or equal to the acceptable flowspec.

Connection request also includes a commitment level that expresses the degree of certainty that the QoS requested will be provided. Three commitment levels are currently defined: guaranteed (QoS will be kept as long as the service, provider functions properly), predicted (QoS will be kept under the assumption that the future load of the service provider corresponds to the load currently observed, resources reserved can be preempted, if necessary, to establish new guaranteed-commitment connections) and best-effort (QoS will be kept as good as possible however, there is no guarantee that the required performance is ever provided and hence, there is no admission control nor resource reservation).

These D-IPC QoS characteristics are directly connected to the ATM network ones.

3.4. A Real-time Java ORB

On top of our real-time distributed platform we need an object oriented framework to support mobility of software components. From our point of view, the Java technology is one of the best candidates.

An open source Java Virtual Machine (JVM) has been ported over our real-time ChorusOS platform. It is based on the Japhar package [36]. The runtime environment provided by our JVM port has been enhanced in order to take advantage of the ChorusOS real-time extensions. New features are used by the way of new classes bundled in a specific Java package. New classes have been introduced to create periodic and aperiodic tasks scheduled according to an EDF policy. Java threads are mapped on kernel threads. They inherit from the Thread class but also provide additional methods required to manage the scheduling parameters specific to our EDF class. As our new framework remains compatible with the original priority-based scheduling classes shipped with Chorus, regular Java threads, simply characterized by a priority level, are scheduled according to a FIFO policy. Our design ensures that FIFO threads always exhibit a criticality level lower than EDF tasks.

Java uses monitors to ensure mutual exclusion when the synchronized keyword is used. Monitors are generally implemented using condition variables based on thread semaphores. In our customized JVM, the implementation of the monitors now relies on our modified real-time mutexes. Consequently, they take advantage of the inheritance protocol provided by the extended ChorusOS kernel.

In addition to the scheduling architecture, other extensions have been featured to the Chorus system, and have been integrated to our JVM. The JVM is able to use the real-time communication system described before through the Java Native Interface (JNI). The deadline concept is extended to Java thread communications that use the end-to-end deadline inheritance over ATM networks. Therefore, the JNI interface has been optimized to allow the direct sharing of memory between Java and native code. This feature also grants Java to access memory segments mapped by data acquisition and control devices.

The current implementation of our RT-JVM seems to be efficient [23]. Our Java threads are able to exchange data at a throughput of 97Mb/s, with 0.3% cell loss and 15 ms delay. Similar experiments in the context of MMS like services on top of a Java ORB environment directly over an ATM network with a regular Linux and a standard JVM [39] give a throughput of 82Mb/s, with a 14.1-45% cell loss, 90 ms delay [38]. For a more accurate comparison, we should test our RT-JVM against an off-the-shelf micro-kernel real-time based JVM, for example PERC [29] that runs over different real-time executives.

4. An Internet based Real-Time Distributed System Architecture

The architecture described in section 3 appears to be very efficient but too specific. Users require the reduction of costs, development time and integration time. Our previous QoS based distributed system design highlight the right properties to be encountered in a object oriented real-time distributed platform. Usually CoS is dedicated to DiffServ Internet networks, but this concept can be extended to execution platform and applies to distributed systems and to applications. We describe hereafter how we map the previous architecture on off-the-shelf products.

This approach is handled via a join project between EDF and CNAM-CEDRIC.

4.1. Real-time operating system

We already evaluated [3] the first software layer of this new architecture, real-time operating systems. Chosen products were LynxOS (Lynx Real-Time Systems), pSOS+ (Wind River, formerly Integrated Systems Inc.) and VxWorks (Wind River), because of the users requirements [2] (real-time aspects, perennity, widely-used in industry, compliance to POSIX standards...). Our conclusions are that all of them comply both performance and function requirements. Other aspects are very important: development environment, documentation, hotline, installation and configuration flexibility. We also evaluated these aspects that are usually never considered.

Final operating systems would be pSOS+ or Wind River, they are both preemptive real-time micro-kernels and offer the needed real-time mechanisms like priority inheritance and zero-copy version of the TCP/IP stack.

4.2 Communication Network

ATM is built to deliver native QoS properties. There is no equivalent properties in the Internet stack. The closer approach corresponds to the IntServ profil [7],

[11] and uses mainly the reservation framework based on the RSVP protocol [14], [26]. The constraint is that all nodes in the communication path including ends need to implement the RSVP protocol, moreover, RSVP loads the network with periodic reservation messages. We decided to eliminate the IntServ profil.

The DiffServ [6], [27] approach is the last available Internet profile. It is not based on QoS handling but on prioritization mechanisms. This approach is called Classification of Service. Diffserv is well-suited to operator networks. We can wonder if it's efficient for our application domain. We can only use the management of priority in the Internet stack for plant communication networks, mapping IP priorities from the TOS field of the datagram into layer 2 frame header.

This is enabled by the 802.1p facility included in the 802.1Q standard (VLAN) [8].

4.3. Java Distributed Objects

There have been different initiatives to promote the use of Java in real-time systems [18] [37] [31]. The most advanced project seems to be PERC [29], a commercial product proposed by NewMonics Inc. PERC uses an extended dialect of Java to enable the specification of execution constraints in the Java code. It also provides a JVM that offers rate monotonic scheduling, priority inheritance, real-time garbage collection and support for runtime execution analysis. Interactions between real-time Java programs and their environment use proprietary Java classes that enable access to I/O ports and interrupt handling. The stringent need for a clear definition of a real-time Java environment and for a standardization of the related core classes led to the creation of a Real-Time Java Working Group (RTJWG) hosted by the NIST US federal organization and, more recently, of the J Consortium that aims to federate ongoing works in this direction. As of this writing, the members of the RTJWG are working on the specification of core real-time extensions for the Java platform [30]. These extensions would offer to Java applications the basic services that are generally offered by commercial real-time operating systems. They should also provide a foundation upon which more sophisticated higher level real-time capabilities would be constructed as optional profiles.

Remote Method Invocation is a high level programming paradigm for distributed object systems. None of the RT-JAVA specifications deals with Remote Method Invocation. This is an important lack that can be handled by specific developments. Approach that we want to avoid now! An alternate solution implies the use of java sockets, a low level abstraction, and to rely on CoS properties of the underlying communication protocols.

5. Conclusion

Our project to build a QoS aware distributed system for power plant control applications is at the middle step of its roadmap. Currently our experiments enforce the interest of the Java environment, it is a promising open distributed platform considering that our real-time constraints are soft.

The Java-extended ChorusOS-ATM platform [25] appears to be very efficient. It is QoS aware and conforms the RM-ODP/ReTINA model [5] based on binding objects. Building such an architecture with off-the-shelf products seems to be difficult today. Off-the-shelf products imply a Classification of Service aware architecture. The gap between QoS and CoS based real-time distributed systems is related to the fact that no off-the-shelf products handle deadlines.

Real-time micro-kernels are generally implemented efficiently, and satisfy the CoS profile. The missing building blocks today are: operating systems don't handle priority in the network stack (priority fields from 802.1p frame and from the TOS field of the IP datagram header). There is no RT-JAVA standard (two concurrent specifications) but PERC from Newmonics [29] is available RT-JAVA specifications don't deal with inter-objects communications such as RMI. CORBA Real-time specifications [33] are more out-of-date and for example support priority propagation, needed by our platform. Despite the interest of a QoS nor CoS Java based distributed system approach, one can say that there is no available off-the-shelf technology today for real-time distributed applications.

Currently, we are porting a JVM (Embedded Java from Sun) on pSOS+ and we want to use the Jonathan ORB on top of it. Jonathan [13] is a RM-ODP compliant ORB that supports a Java-RMI personnality. There also exists a QoS implementation in Jonathan, based on the RSVP protocol [4]. We think such a platform, a JVM with real-time extensions on top of a real-time micro-kernel will be available as an off-the-shelf soon. All of the real-time micro-kernels vendors are already proposing a JVM based offer (without real-time capabilities). Also, we will experiment the advantages of Jini [42] for platform deployement and configuration.

References

[1] L. Bacon, E. Becquet, E. Gressier-Soudan, C. Lizzi, C. Logé, L. Réveilleau. "Provisioning QoS in Real-Time Distributed Object Architectures for Power Plant Control Applications". *2nd IEEE International Symposium on Distributed Objects and Applications. DOA'00.* September 21-23, 2000. Antwerp. Belgium.

[2] L. Bacon, E. Gressier-Soudan. *SETR : Systèmes d'Exploitation Temps Réel - Etude des contraintes utilisateurs – Synthèse entretiens*. HP-32/98/079/A. 1999.

[3] Bacon, E. Becquet, E. Gressier-Soudan. "Etude Comparative de micronoyaux du commerce". *7ème Séminaire du Laboratoire d'Ingénierie de la Sûreté de Fonctionnement.* 16-17 Mars 1999. Domaine de Villepreux. Bordeaux. France.

[4] Becker. *"Qos in Jonathan with RSVP"*. 1998.

[5] Blair, J-B. Stefani. *"Open Distributed Processing and Multimedia"*. Addison-Wesley. 1997.

[6] S. Blake et al. "An Architecture for Differentiated Services". *RFC 2475, IETF DiffServ Working Group*. December 1998.

[7] B. Braden, S. Shenker, and D. Clark. "Integrated Services in the Internet Architecture: an Overview". *RFC 1633, IETF IntServ Working Group*. June 1994.

[8] R. Breyer, S. Riley. *"Switched, Fast and Gigabit Ethernet, Understanding, Building, and Managing High-Performance Ethernet Networks"*. 3rd Edition. MacMillan Technical Publishing. Macmillan Network Architecture and Development Series. 1999.

[9] Buttazzo, J.A. Stankovic. RED: A Robust Earliest Deadline Scheduling Algorithm. *3rd International Workshop on Responsive Computing Systems*, September 1993.

[10] Chevassus. P. Curnier. R. Gaches. E. Gressier. S. Natkin. C. Toinard. *"Pilot Requirement Definition - Aerospatiale Demonstration Facility"*. Esprit Project 7096. CIME Computing Environment Integrating a Communication Network for Manufacturing Applications (CCE-CNMA). Deliverable 3. Workpackage 7. July 1993. Editor: AEROSPATIALE. CCE-CNMA Consortium. Private Access Document.

[11] D. Clark, S. Shenker and L. Zhang. *"Supporting Real-Time Applications in an Integrated Services Packet Network : Architecture and Mechanisms"*. ACM Sigcomm Proc. 1992.

[12] CS Technologies Informatiques. *"ANTARA System Functional Specification"*. Document 101-00061, CSTI, April 1996.

[13] B. Dumant, F. Dang Tran, F. Horn, and J.-B. Stefani. "Jonathan: an open distributed processing environment in Java". In N. Davies, K. Raymond, and J. Seitz, editors, *Middleware'98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, The Lake District, U.K., September 1998.

[14] G. Gaines and M. Festa. "A survey of RSVP/QoS Implementations", update 2, *RSVP Working Group*. July 1998.

[15] L. George, C. Lizzi, J. Montiel. "External/Internal Clock Synchronization in ATM-based Distributed Systems". *23 rd Euromicro Conference*, September 1997.

[16] E. Gressier-Soudan, M. Epivent, A. Laurent, R. Boissier, D. Razafindramary, M. Raddadi, "Component oriented control architecture, the COCA project" *Special Issue on Manufacturing, Microprocessors and Microsystems Journal* - V23N2 September 1999, p95-102, Elsevier Science.

[17] R. Grimes. Professionnal *"DCOM Programming, A guide to creating practical applications with Microsoft's Distributed Component Object Model"*. Wrox Press. 1997.

[18] J consortium. *"Real-time core extensions specifications for the Java platform"*. February 2000.

[19] G. Koren, D. Shasha. " An Optimal On-Line Scheduling Al-gorithm for Overloaded Real-Time Systems". *13IEEE Real-Time Systems Symposium*, December 1992.

[20] K. Kusunoki, I. Imai, H. Ohtani, T. Nakawaji, M. Ohshima, K. Ushijima. "A CORBA-Based Remote Monitoring System for Factory Automation". *Proceedings of the 1st IEEE International Symposium on Object Oriented Real-Time Distributed Computing. ISORC'98*. Kyoto Japan. April 20-22. 1998.

[21] C. Lizzi, E. Gressier-Soudan, J. Montiel. "A Real-Time Communication Service for ATM-based Distributed Systems". *ICATM'98. IEEE International Conference on ATM.* June 98. Colmar France.

[22] C. Lizzi, E. Gressier-Soudan. "A Real-Time IPC Service over ATM networks for the Chorus Distributed System". *Euromicro'98. 24th Euromicro Conference*. Vasteras. Sweden. August 1998.

[23] C. Lizzi. "Java Real-Time Distributed Processing over ATM Networks with ChorusOS". *ETFA'99*. Barcelona, Catalogna. Spain. October 1999.

[24] C. Lizzi. "Enabling Deadline Scheduling for Java Real-Time Computing". *20th IEEE Real Time Systems Symposium (RTSS'99).* Phoenix, Arizona, USA. December 1999.

[25] C. Lizzi. *"Design of an ATM based Real-Time Distributed System"*. PhD Thesis. Conservatoire National des Arts et Métiers. 14 Décembre 1999. Paris. France. (text in french).

[26] A. Mankin, ed., "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement : Some Guidelines on Deployment". *RFC 2208, IETF RSVP Working Group.* September 1997.

[27] K. Nichols, V. Jacobson, and L. Zhang. *"A two-bit Differentiated Services Architecture for the Internet"*. November 1997.

[28] I. Mizunuma, S. Chia, M. Takegaki, "New architecture of industrial systems with real-time ATM middleware". *3rd IEEE International Workshop on Real-Time Computing Systems and Applications*. 1996.

[29] K. Nilsen, *"Issues in the Design and Implementation of Real-Time Java"*, Technical Report, NewMonics Inc., 1996.

[30] K. Nilsen, *"Functional Requirements for Core Real-Time Extensions for the Java Platform"*, Draft, Real-Time Java Working Group, May 1999.

[31] National Institute Of Standards and Technology (NIST). *"Requirements For Real-time Extensions for the Java platform"*. September 1999.

[32] http://www.objectweb.org/. March 31st.2000.

[33] Object Management Group. *"Real-Time CORBA"*. Joint Revised Submission. March 1999.

[34] A. Polze, D. Plakosh, K. Wallnau. "CORBA in Real-Time Settings: A problem from the Manufacturing

Domain". *Proceedings of the 1st IEEE International Symposium on Object Oriented Real-Time Distributed Computing. ISORC'98*. Kyoto Japan. April 20-22. 1998.

[35] http://www.PROFIBUS.com/, March 24th 1999

[36] P. Reinholdtsen. *"Japhar : the Hungry Programmers Open Source Java"*. Norwegian Unix Users Group. November 1998.

[37] The Real-time Java Experts Group. *"Real-time Specifications for Java"*. February 2000.

[38] L. Seinturier. *"Intégration de liaisons ATM dans l'ORB CORBA Jonathan"*. Personnal Communication. January 1999.

[39] L. Seinturier, A. Laurent, B. Dumant, E. Gressier-Soudan, F. Horn. "A Framework for Real-Time Communication Based Object Oriented Industrial Messaging Services". *ETFA'99*. Barcelona, Catalogna. Spain. October 1999.

[40] J. Stankovic, M. Spuri, K. Ramamritham, G. Buttazzo. *"Deadline Scheduling for Real-Time Systems, EDF and Related Algorithms"*. Kluwer Academic Publishers. 1998.

[41] J-B. Stefani. *"Requirements for a real-time ORB"*. OMG Contribution. May 1996.

[42] Sun Microsystems, *"Jini Architecture Specification"*, Technical Report, January 1999.

[43] Valenzano. Demartini. Ciminiera. *"MAP and TOP Communications"*. Addison Wesley. 1992.

[44] http://WWW.worldfip.org/faq.html, March 24th 1999.