

The seal of the University of Bologna is a large, circular emblem in the background. It features a central figure, likely a saint or scholar, surrounded by various scenes and text. The outer ring contains the Latin motto "S. PAVS UBIVQV PATEP" and "AN 1088". The inner ring contains "COLL-JUR-PATI", "COLL-MED", "COLL-JUR-CIVIL", and "ET-ART".

Jgroup: Enhancing Jini with Group Communication

Alberto Montresor

Renzo Davoli

Özalp Babaoglu

Technical Report UBLCS-2000-16

December 2000

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)

The University of Bologna Department of Computer Science Research Technical Reports are available in gzipped PostScript format via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` or via WWW at URL `http://www.cs.unibo.it/`. Plain-text abstracts organized by year are available in the directory `ABSTRACTS`. All local authors can be reached via e-mail at the address `last-name@cs.unibo.it`. Questions and comments should be addressed to `tr-admin@cs.unibo.it`.

Recent Titles from the UBLCS Technical Report Series

- 99-17 *A Simple Game Semantics Model of Concurrency*, Asperti A., Finelli M., Franco G., Marchignoli D., July 1999.
- 99-18 *A Complete Axiomatization for Observational Congruence of Prioritized Finite-State Behaviors*, Bravetti, M., Gorrieri, R., July 1999.
- 99-19 *Middleware for Dependable Network Services in Partitionable Distributed Systems*, A. Montresor, R. Davoli, O. Babaoglu, October 1999 (Revised April 2000).
- 99-20 *Performance Analysis of Software Architectures via a Process Algebraic Description Language*, Bernardo, M., Ciancarini, P., Donatiello, L., November 1999 (Revised March 2000).
- 99-21 *Real-Time Traffic Transmission Over the Internet*, Furini, M., Towsley, D., November 1999.
- 99-22 *On the Expressiveness of Event Notification in Data-Driven Coordination Languages*, Busi, N., Zavattaro, G., December 1999.
- 2000-1 *Compositional Asymmetric Cooperations for Process Algebras with Probabilities, Priorities, and Time*, Bravetti, M., Bernardo, M., January 2000 (Revised February 2000).
- 2000-2 *Compact Net Semantics for Process Algebras*, Bernardo, M., Busi, N., Ribaudo, M., March 2000 (Revised December 2000).
- 2000-3 *An Asynchronous Calculus for Generative-Reactive Probabilistic Systems*, Aldini, A., Bravetti, M., May 2000 (Revised September 2000).
- 2000-4 *On Securing Real-Time Speech Transmission over the Internet*, Aldini, Bragadini, Gorrieri, Rocchetti, May 2000.
- 2000-5 *On the Expressiveness of Distributed Leasing in Linda-like Coordination Languages*, Busi, N., Gorrieri, R., Zavattaro, G., May 2000.
- 2000-6 *A Type System for JVM Threads*, Bigliardi, G., Laneve, C., June 2000.
- 2000-7 *Client-centered Load Distribution: a Mechanism for Constructing Responsive Web Services*, Ghini, V., Panzieri, F., Rocchetti, M., June 2000.
- 2000-8 *Design and Analysis of RT-Ring: a Protocol for Supporting Real-time Communications*, Conti, M., Donatiello, L., Furini, M., June 2000.
- 2000-9 *Performance Evaluation of Data Locality Exploitation* (PhD Thesis), D'Alberto, P., July 2000.
- 2000-10 *System Support for Programming Object-Oriented Dependable Applications in Partitionable Systems* (PhD Thesis), Montresor, A., July 2000.
- 2000-11 *Coordination: An Enabling Technology for the Internet* (PhD Thesis), Rossi, D., July 2000.
- 2000-12 *Coordination Models and Languages: Semantics and Expressiveness* (PhD Thesis), Zavattaro, G., July 2000.
- 2000-13 *Jgroup Tutorial and Programmer's Manual*, Montresor, A., October 2000.
- 2000-14 *A Declarative Language for Parallel Programming*, Gaspari, M., October 2000.
- 2000-15 *An Adaptive Mechanism for Securing Real-time Speech Transmission over the Internet*, Aldini, A., Gorrieri, R., Rocchetti, M., November 2000.
- 2000-16 *Jgroup: Enhancing Jini with Group Communication*, Montresor, A., Babaoglu, O., Davoli, R., December 2000.

Jgroup: Enhancing Jini with Group Communication

Alberto Montresor¹

Renzo Davoli¹

Özalp Babaoğlu¹

Technical Report UBLCS-2000-16

December 2000

Abstract

Reliable group communication is an important technology for building fault-tolerant applications. Yet, some of the major frameworks for distributed application development (e.g., Corba, Jini and Enterprise JavaBeans) lack support for this paradigm. We claim that this situation constitutes a major obstacle to a broader diffusion of reliable group communication in industrial applications. In this paper, we discuss some of the main issues related to integrating reliable group communication and Jini.

1. Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, Bologna 40127 (Italy). Tel: +39 051 2094871, Fax: +39 051 2094510. Email: {montresor,davoli,babaoğlu}@CS.UniBO.IT

1 Introduction

Reliable group communication is an important technique to build fault-tolerant applications that consistently maintain replicated state of some sort [8, 3, 15, 9, 1]. Unfortunately, industrial applications using this paradigm are few. In our opinion, the reason behind the lack of acceptance in the industry is that important frameworks for distributed application development such as Corba [6], Jini [2] and Enterprise JavaBeans [12] do not include any support for reliable group communication.

In all these frameworks, support for fault-tolerance is limited to the *transactional paradigm*, in which operations performed on one or more objects may be grouped together in order to make a set of guarantees on their execution. An example of such guarantees is atomicity (i.e., *all-or-nothing*): all operations in a transaction are executed correctly, or none is.

In the CORBA model, however, the situation is going to change: together with the *object transaction service* (OTS) [5], Corba developers will be able to use the new *fault-tolerant corba service* (FT-Corba) [7]. Market products implementing FT-Corba are expected to be shipped at the beginning of 2001. FT-Corba is aimed at providing support for the development of replicated object groups, insulating clients of such groups from details such as group management, reliable group communication, failure masking and recovery.

1.1 The Problem

In a recent paper [4], Frolund and Guerraoui criticize the fact that the FT-Corba proposals consider replications and transactions as separate aspects of fault tolerance. As a result, the composition of FT-Corba and OTS does not result in any meaningful combination of their respective strengths. The problem lies on the fact that these services have been specified as coarse-grained abstractions, thus precluding the possibility of developing Corba-compliant replication and transaction services which are both replication-aware and transaction-aware. The authors advocate an approach where the fundamentals building blocks of these services are standardized.

As an example, consider the OTS specification. Each transaction is conceptually driven by a *transaction coordinator*. Coordinators are based on a “crash-restart” recovery model, where transaction participants may remain blocked until their coordinator recovers. In many situations, replication-based fail-over would be preferable. Unfortunately, using FT-Corba to replicate coordinators is not straightforward, as there is no access to the coordinator definition through the standardized OTS API.

1.2 Contribution

Corba is becoming the most important industrial standard to build distributed applications. Corba, however, is not the only framework for developing such applications; notable alternatives are Jini [2] and Enterprise JavaBeans [12]. In order to promote the utilization of group communication in distributed applications, we believe that this paradigm should be integrated in these frameworks as well.

In this paper, we present Jgroup [14], an object group system which enhances the Jini model with group communication. Aim of Jgroup is to enable the construction of replicated Jini services capable to be federated in a standard Jini distributed system. As in Corba, this requires not only the presence of an object group service, but also a strict integration between the transactional model promoted by Jini and the group communication model. Differently from Corba, this integration is simpler, as the Jini transaction specification has been designed in order to precisely identify the different roles that objects may assume in transactions, such as transaction managers, participants and clients.

The rest of the paper is organized as follows. In Section 2, we provide a short introduction to Jini. In Section 3, we discuss how Jgroup integrates reliable group communication and Jini. Finally, conclusions are drawn in Section 4, together with directions for future work.

2 Jini

The most important concept within the Jini architecture is that of a *service*, a software component providing some facility. Services appear programmatically as objects written in Java, with an interface that defines the operations that can be requested on them. The aim of Jini is to federate services and service clients into a single, dynamic distributed system. Services can join the distributed system in a robust fashion, and clients can rely on the availability of visible services, or at least upon clear failure semantics.

Apart from services, the components of the Jini architecture may be divided in two other categories: *infrastructure* and *programming model*. The infrastructure is the set of components that enables building a federated Jini system, and defines the minimal Jini core. The infrastructure is composed of the Java RMI protocol [11], which enables objects to communicate through remote method invocations, and the lookup service [2], which provides a central registry for services. Java RMI is based on the concept of *proxy*, which is the local representative of the remote service being accessed by a client. Proxies implement the same remote interfaces as their remote counterparts and take care of all low-level details of communication between clients and servers. The lookup service enables the registration of proxies for services, together with information about the service type and a collection of attributes. Client may retrieve services by specifying a type and (possibly) a set of attributes for matching.

The Jini specification defines the programming model as a set of interfaces that enables “the construction of reliable services”, including those that are part of the infrastructure (as the lookup service) and those that join into the federation. The programming model is based on three distinct paradigms for distributed computing: *events*, *leases* and *transactions*. The event notification interfaces enable event-based communication between Jini services, extending the event model used by JavaBeans components to the distributed environments.

The lease interface extends the Java programming model by adding the time to the notion of holding a reference to a resource, enabling references to be reclaimed safely in the face of partitions. As an example, registrations in the lookup service are leased: a service must periodically renew its registration, otherwise its proxy is removed when its lease expires.

The transaction interfaces introduce a lightweight, object-oriented protocol enabling Jini services to coordinate state changes. The Jini transaction protocol differs from existing transaction interfaces (e.g., those defined in Corba) in that it does not assume that transactions follow a particular transaction semantics. Jini takes a more object-oriented view, leaving the correct implementation of transaction semantics up to the designer of the object involved in the transaction. The Jini transaction specification has identified the basic components of a transaction, such as *transaction clients*, *managers*, and *participants*. Transaction clients start a transaction by contacting a transaction manager through a proxy. The proxy is obtained by querying the lookup service for a service implementing the manager interface. The transaction manager responds with a *semantic transaction object*, which will represent the transaction in the following communications and contains information such as an identifier for the transaction and the proxy for the transaction manager. At this point, clients interact with participants by communicating the semantic object and the operation requested. The participants use the semantic object to communicate their commit/abort vote to the transaction manager. The transaction manager oversees the consistent execution of the operations and guarantees that all participants will eventually know if they should commit the operations or abort them. All interactions between clients, managers and participants are based on Java RMI.

In contradiction with the claims of the Jini specification, the Jini architecture does not provide an adequate support for the development of reliable and high-available distributed applications. In particular, no support is provided for implementing a service as a group of replicated objects, nor by the architecture neither from the programming model.

3 Integrating Jgroup and Jini

The Jgroup middleware system [13, 14], currently under development at the University of Bologna, is an object group communication service written in Java. Jgroup is being integrated with Jini in order to provide a more suitable environment for the development of reliable services. The aim of this section is to describe the main characteristics of this integration and discuss some of the open issues.

Jgroup is based on the *object group* paradigm. In this paradigm, distributed services that are to be made dependable are replicated among a dynamic collection of server objects that implement the same set of remote interfaces and form a group in order to coordinate their activities and appear to clients as a single service. Clients access a replicated service by interacting with a *group proxy*, indistinguishable from a standard proxy used to access non-replicated services. Group proxies handle all low-level details such that clients need not be aware that the service is being provided by a group rather than a single server object. In particular, clients are unaware of the number, location or identity of individual servers in the group. Communication between clients and groups, and among servers composing the group takes the form of *reliable group method invocations*, that result in methods being executed by one or more servers forming the group, depending on the invocation semantics.

The integration between Jini and the object group communication model provided by Jgroup has to be performed at each level of the Jini architecture, thus involving infrastructure, programming model and services. First of all, we have extended the Java RMI protocol [11]. Unfortunately, the current Java RMI API does not enable the protocol to be easily expanded. On the client side, RMI proxies contain *remote references*, which maintain the information needed to locate their remote counterparts and implement the communication protocol on behalf of proxies. Java RMI enables the use of customized remote references; nevertheless, there is no analogous customizable entity on the server side. This lack makes remote references completely useless for group communication, as they enable the modification of the communication protocol only on one side. In order to avoid this limitation, we have had to modify the standard `rmic` compiler which is responsible for the generation of stubs and skeletons classes (the client- and the server-side proxies for Java RMI, respectively). We have added an additional communication layer on both sides, that deals with all details of group invocations, such as reliability, eliminations of duplicates and result selection.

The Jini lookup service has required modification for dealing with group proxies. The reference implementation of the lookup service enables the registration of customized proxies for services. This feature could be used to register group proxies using any implementation of the lookup service. Group proxies, however, differ from standard proxies as their contents may be dynamic. A server registering in a lookup service must not overwrite existing information about previously registered group servers. Instead, it must add its information to an existing group proxy. Furthermore, when a server crashes or becomes partitioned, and fails to renew the lease obtained from the lookup service when registering, the information about it has to be removed from the group proxy, clearly without removing the entire proxy. These considerations lead us to develop an alternative implementation of the lookup service, in which group servers register their information in a group proxy by specifying a group name attribute.

In order to integrate group communication with the transactional model, we are following a two-step approach. First, we have implemented a replicated transaction manager using the group communication facilities of Jgroup. For clients and participants, the presence of a replicated transaction manager is completely transparent: clients obtain a group proxy for the transaction manager by querying the lookup service for a service implementing the manager interface, as in the non-replicated case. As noted before, group proxies are completely undistinguishable from standard RMI proxies (both of them simply implement the remote interfaces they represent). Clients starts a transaction by invoking a method on the group proxy, and obtain a semantic transaction object. Participants are notified by clients about transactions, together with information about the transaction manager enclosed in the semantic object. Following the Jini specification, participants interact with the transaction manager through the semantic object. The semantic

object (actually, the group proxy for the manager included in it) hides to participants the fact that the transaction manager is replicated. Having a replicated transaction manager guarantees a higher availability of the transaction service than "crash-recovery" model offered by the reference implementation of the Jini transaction specification.

In this first step, we have considered only the role of transaction manager as a candidate for replication. In the second step, we are completing the integration between group communication and transactions by enabling each of the roles identified in the Jini specification to be performed by a group of replicated objects. For example, an object group acting as a client may contact a replicated transaction manager, in order to request a set of operations to be performed by a collection of services. Some of these services could be non-replicated objects, while other could be object groups. As before, the aim is to provide complete transparency for non-replicated entities. This means that non-replicated participants should receive a single request from replicated clients, while non-replicated clients should not be required to be aware that they are requesting an operation to be performed by an object group. Once again, the elimination of duplicated requests and the enforcing of the reliability guarantees is performed by proxies.

4 Concluding Remarks and Future Work

Integrating reliable group communication with existing distributed development frameworks such as Corba [6], Jini [2] and Enterprise JavaBeans [12] is necessary to promote a broader diffusion of the group communication paradigm in the industry. In this paper, we have described how an object group toolkit could be integrated with Jini. The transactional interfaces of Jini are well-suited to be integrated with group communication, as they precisely identify the fine-grained roles that may be performed by objects in a transaction, without specifying a monolithic service as in Corba.

Jgroup is the subject of a collaborative research project between Sun Microsystems and the University of Bologna. The first result of this collaboration is the definition of a new Java RMI API that will enable users to plug in custom behaviors in the Java RMI protocol [10]. In particular, the new RMI API enables developers to customize both the client and the server side, making it suitable for implementing the reliable group method invocation semantics of Jgroup.

In their paper [4], Frolund and Guerraoui argue that the combination of the replication and transactional services included in Corba is not adequate for the development of reliable applications for the kind of architectures for which Corba is considered successful, i.e. pure three-tier architectures. We plan to exploit the experience gained putting together Jgroup and Jini in order to integrate the group communication paradigm also in the Enterprise JavaBeans model [12], which explicitly promotes the development of applications based on the three-tier architecture.

References

- [1] T. Anker, G. Chockler, D. Dolev, and I. Keidar. Fault-Tolerant Video-on-Demand Services. In *Proc. of the 19th Int. Conf. on Distributed Computing Systems*, pages 244–252, June 1999.
- [2] K. Arnold, B. O'Sullivan, R. Scheifler, J. Waldo, and A. Wollrath. *The Jini Specification*. Addison-Wesley, 1999.
- [3] R. Friedman and A. Vaysburg. Fast Replicated State Machines over Partitionable Networks. In *Proc. of the 16th Symp. on Reliable Distributed Systems*, October 1997.
- [4] S. Frolund and R. Guerraoui. Corba Fault-Tolerance: why it does not add up. In *Proc. of the 1999 IEEE Workshop on Future Trends in Distributed Computing (FTDCS)*, Capetown, December 1999.
- [5] Object Management Group. CORBA Services – Transaction Service. Technical report, Object Management Group, Framingham, Ma, November 1997.
- [6] Object Management Group. *The Common Object Request Broker: Architecture and Specification, Rev. 2.3*. OMG Inc., Framingham, Mass., March 1998.
- [7] Object Management Group. Revised Joint Fault Tolerance Submission. Technical report, Object Management Group, Framingham, MA, 1999.

- [8] I. Keidar and D. Dolev. Efficient Message Ordering in Dynamic Networks. In *Proc. of the 15th ACM Symp. on Principles of Distributed Computing*, Philadelphia, PA, May 1996.
- [9] R. Khazan, A. Fekete, and N. Lynch. Multicast Group Communication as a Base for a Load-Balancing Replicated Data Service. In *Proc. of the 12th Symp. on Distributed Computing*, August 1998.
- [10] Sun Microsystems. JSR 78 - RMI Custom Remote Reference. Url:
http://java.sun.com/aboutJava/communityprocess/jsr/jsr_078_rmicrr.html.
- [11] Sun Microsystems. *Java Remote Method Invocation Specification, Rev. 1.50*. Sun Microsystems, Inc., Mountain View, California, October 1998.
- [12] Sun Microsystems. *Enterprise JavaBeans Specification, Version 1.1*. Sun Microsystems, Inc., Mountain View, California, December 1999.
- [13] A. Montresor. *System Support for Programming Object-Oriented Dependable Applications in Partitionable Systems*. PhD thesis, Dept. of Computer Science, University of Bologna, February 2000.
- [14] A. Montresor, R. Davoli, and Ö. Babaoğlu. Middleware for Dependable Network Services in Partitionable Distributed Systems. In *Proc. of the First PODC Workshop on Middleware*, Portland, Oregon, July 2000.
- [15] J. Sussman and K. Marzullo. The *Bancomat* Problem: an Example of Resource Allocation in a Partitionable Asynchronous System. In *Proc. of the 12th Symp. on Distributed Computing*, 1998.