Implementing FT-CORBA with Portable Interceptors: Lessons Learned

Fabíola Greve[⋆†] Jean-Pierre Le Narzul[•†]
{fgreve|jlenarzu}@irisa.fr

⋆ LaSiD-DCC-UFBA
Campus de Ondina
40170-110 Bahia, Brasil

† IRISA
Campus de Beaulieu
35042 Rennes, France

• ENST-Bretagne
Campus de Rennes
35512 Cesson-Sévigné, France

Abstract

In this paper, we present the design of Open EDEN, an implementation of the FT-CORBA specification based on the use of a group communication framework, called EDEN. The design of Open EDEN was driven by the desire to use only portable techniques (mainly portable interceptors) to integrate the EDEN framework within a CORBA platform. We discuss the main difficulties we encountered and we draw some conclusions about the adequacy of this choice.

1 Introduction

Fault tolerance in CORBA (Common Object Request Broker Architecture) is provided through object replication. A critical CORBA service can be made reliable by replicating a CORBA object implementing it on different sites of the distributed system. The set of replicas forms an object group. If sites fail independently, an invocation to the service will succeed even if some replicas of the group have crashed.

Apart redundancy, fault tolerance in CORBA relies also on fault detection and notification, logging and recovery. All the components of a FT-CORBA architecture are described in an OMG specification [5] which has been recently adopted by the OMG board. The specification describes the interfaces of the services, different fault tolerance strategies and fault tolerance properties associated to an object group.

The designer of a FT-CORBA architecture is left with a lot of freedom in choosing the way the components of this architecture are implemented and the way they interact. Consequently, there have been several efforts to add fault tolerance in CORBA systems. Some of these efforts adhere very closely to the FT specification ([4, 7, 1]); others are voluntarily less FT-CORBA compliant and favor innovative solutions ([2]). Among these efforts, we can distinguish three major approaches [4]: (i) the service approach in which clients and servers must explicitly interact with a specific service to get reliability; (ii) the integration approach in which a group communication framework providing replication functionnality is integrated inside the ORB; (iii) the interception approach in which, requests or messages (depending on the location of the interception layer) are intercepted and redirected to a group communication framework.

Of course, each of these approaches presents advantages and drawbacks. The service approach is probably the simplest to implement but has a major drawback that is the lack of transparency. The integration approach is the most efficient but cannot be easily re-used between ORBs as it is very dependent on ORB internals. The interception approach offers transparency but, depending on how it is implemented, can be non-portable (from a system point of view) if the interception layer is placed between the ORB and the system interface.

We are currently prototyping an architecture exclusively based on the interception approach for supporting active replication of CORBA objects. For this development, we decided to use only portable techniques. By portable, we mean (1) portable with respect to the system support; (2) portable with respect to the group communication system; (3) portable with respect to the ORB. The first condition precludes the use of system-level interceptors like in Eternal [4]. The second condition precludes a tightly coupling between some components of our FT architecture and the group communication system. The third condition leads us to use a specific feature of CORBA which are portable interceptors [6]. Portable interceptors are the only portable mechanism specified by the OMG which allows to transparently add service code to the ORB. The objective of this paper is to describe this experience in using portable interceptors for prototyping a fault-tolerant CORBA system and to draw some conclusions about the adequacy of this technique. Our prototype is based on a group communication system called EDEN.

The remainder of this paper is structured as follows. Section 2 briefly describes the EDEN group communication framework. Section 3 summarizes the FT-CORBA specification and introduces CORBA portable interceptors. Section 4 presents our prototype which implements a FT-CORBA service based on the use of the EDEN framework. Section 5 lists some of the problems we encountered. Section 6 concludes this paper.

2 The EDEN Group Communication Framework

A group service allows a collection of related objects to be considered externally as a single logical entity. This paradigm is used to simplify replication management and construct fault-tolerant applications. Because of its importance, many group communication frameworks have been constructed [13]. The group service includes different primitives used by group members to coordinate their activities. One of those primitives is called *atomic broadcast*, which assures that all requests received by the group entity are delivered to its members in the same order. Another fundamental abstraction is the *group membership*. This service tracks changes in the group composition, due to the desire of members to join or leave the group, or the occurrence of crashes. The *view synchrony communication* allows the delivery of requests “in synchrony” with the delivery of changes in the group composition. All these abstractions are considered as agreement problems that can be solved as a reduction to the *consensus* problem [8, 9].

EDEN is a configurable group-based toolkit, which aims at developing reliable, object-oriented, distributed applications. It is formed of two components: a library of protocols (named ADAM), and an event-based architecture for structuring services (named EVA [12]).

ADAM is a library of autonomous agreement protocols which can be pieced together to implement reliable services. At the heart of this library is a General Agreement Framework (GAF) [9] suited to generate ad hoc agreement protocols. GAF is a Chandra-Toueg [8] consensus-based service which implements some versatility parameters that encompasses many interesting features (early decision, multiple propositions of initial values, decision on a vector of values, deferring of initial values proposals) suitable to generate efficient protocols. In order to solve a particular agreement protocol one must instantiate the parameters with appropriate values.

Clever instantiations of the GAF functions have been investigated in order to implement the group membership [11], atomic broadcast and view synchrony communication protocols.

The framework EVA [12] implements a publish-subscriber communication environment to structure entities composing high level protocols. In this architecture, protocols are regarded as a number of cooperating objects (entities) that communicate via an event channel.

3 CORBA

3.1 FT-CORBA Standard

The FT CORBA specification defines a set of standard interfaces for replication management, fault management and logging/recovery management. One of the main components of the FT-CORBA architecture is the replication manager which interfaces allow to specify the fault tolerance properties of an object group. The replication style is one of this property for which three values are possible: `COLD_PASSIVE`, `WARM_PASSIVE` (passive replication is based on the use of a primary replica) and `ACTIVE` (all replicas play the same role in the group).

One important point in the specification is the structure of an object group reference. Such reference must be built according a standard schema in order to allow inter-operability between client and server hosted by heterogeneous infrastructures. The specification defines the use of multiple `TAG_INTERNET_IOP` profiles to identify replicas (or a set of gateways) and introduces a new component `TAG_FT_GROUP` to encapsulate an object group identifier. An object group is identified by a Group IDentifier (GID) which does not depend of the group membership. At group creation, the replication manager allocates the GID for the group and then constructs an object group reference populated with this GID.

The specification does not define a standard multicast group communication protocol to be used for maintaining consistency between replicas. Consequently, there is no inter-operability between ORBs hosting server objects ; all the replicas of a server object group must be hosted by the same FT infrastructure. This particularity allows FT CORBA designers to choose the most appropriate protocol for their infrastructure.

3.2 Portable Interceptors

Portable interceptors are hooks into the ORB through which ORB services can intercept the normal flow of execution of the ORB. The specification [6] defines IOR (Interoperable Object Reference) interceptors and request interceptors. The former allows to add specific components inside a IOR during its creation. The latter allows to intercept the flow of a client invocation to a server at different points; some of these points are located at the client side and others are located at the server side. A client request interceptor allows to execute service code before the request is sent to the server and before the reply is returned to the client code. Using a specific CORBA exception mechanism, the interceptor code can change the target of an invocation. A server request interceptor allows to execute service before delivering the request to the server object and after the server object has completed its job.

4 Prototyping an EDEN Based FT-CORBA Service

4.1 Overall Architecture

As mentioned in the introduction of this paper, our goal is (1) to design a portable and flexible fault-tolerant service for CORBA and (2) to experiment the usability of EDEN as a group communication support for such a service. For the sake of simplicity and because we did not focus on a full FT-CORBA prototype, we decided the following restrictions:

- We consider only active replication (`ReplicationStyle` property set to `ACTIVE`). Consequently, logging and recovery mechanisms (only needed for passive replication) are not implemented.
- Membership and consistency are controlled by the infrastructure (`MemberShipStyle` and `ConsistencyStyle` properties are respectively set to `MEMB_INF_CTRL` and `CONS_INF_CTRL`).
- Fault detection is not accessible via CORBA interfaces.

Figure 1 shows the main components of our architecture (named Open EDEN) : (1) a gateway [1] which acts as an intermediary object on the request path between a client and a server object group; (2) a CRPI component, which is a Client Request Portable Interceptor attached to a client and used to redirect client requests to a gateway; (3) a SRPI component, which is a Server Request Portable Interceptor attached to each server hosting a replica and used to implement the interface with the EDEN framework.

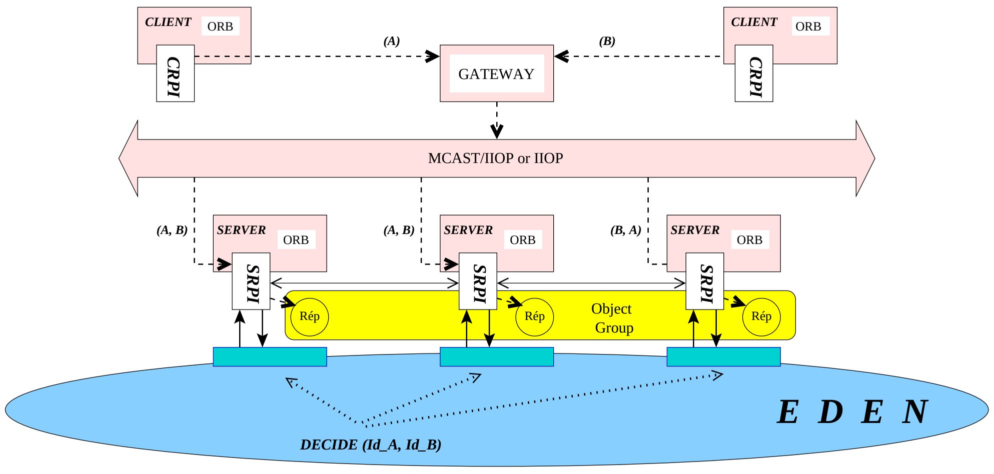

Figure 1: Overview of the Open EDEN Architecture

4.2 Client Redirection

The client of a server object group obtains the object group reference through usual ways, i.e. either by using a call to the CORBA Naming Service or by reading it from a file. Then, the

[1]We only represent one gateway in the figure but for fault-tolerance reasons, our architecture supports multiple gateways.

client transparently (i.e. without knowing that the target object is an object group) invokes the method of the server object group. At this point, a client interceptor intercepts the call, extracts identification of gateways and the group identifier from a tagged component and finally redirects the request towards one of the available gateways.

As represented in Figure 1, the protocol used by the gateway to address an object group can be either IIOP or MCAST/IIOP. In the former case, the group invocation is accomplished by using multiple unicasts. In the latter, the invocation is multicasted to the group. We will not give details in this paper on how the gateway obtains the multicast reference (or list of references); in our prototype, the mapping between the object group identifier and references is implemented by the replication manager.

4.3 Replica Consistency

Infrastructure-controlled consistency is one of the main requirements of the FT-CORBA specification. As stated in the specification, it means that for the `ACTIVE` replication style, ”at the end of each method invocation on the object group, even in the presence of faults, the members should have the same state”. To maintain group consistency, it is necessary that (1) each member behaves deterministically and that (2) the same sequence of requests be delivered in the same order to each member of the group. Regarding the first condition, like Narasimhan in [3], we consider that the unique source of undeterminism is due to potential concurrent executions of threads; we provide a specific scheduler to solve it. For the second condition, thanks to the EDEN framework, we built an adequate delivery protocol. In the following paragraphs, we discuss strategies we used to (1) re-order the requests, (2) to ensure their reliable delivery and (3) to finalize their execution. All these strategies are implemented in a Server Request Portable Interceptor (SRPI) which interfaces CORBA with the EDEN framework.

To minimize the quantity of information exchanged between the two worlds (CORBA and EDEN) and to optimize the performance of the protocols inside the EDEN framework, we decided that the SRPI components will feed EDEN with requests identifiers (instead of whole requests). The consequences of this choice are also examined in the following paragraphs.

Re-ordering requests Each time a client request is intercepted, its thread is blocked and the SRPI component proposes its identifier (request identifier) to the EDEN framework. At each site, EDEN collects the proposed identifiers and runs a consensus protocol with its peers to agree on a set of identifiers and a single delivery order for them. When a decision (or a set of decisions) has been taken, the EDEN framework calls the SRPI components and gives them an ordered list of identifiers; each SRPI component can then mark the corresponding requests as deliverable. To protect against concurrent threads executions, a scheduling algorithm is implemented by each SRPI component to serialize the execution of the deliverable requests by their associated threads.

Reliable delivery Due to the unreliability of the MCAST protocol or the intermediate view of the replication manager, an SRPI component could be asked by the EDEN framework to deliver a request that it has never received. To solve this problem, we added a specific operation to the CORBA-EDEN interface. When the EDEN framework detects that a particular SRPI has not proposed an identifier for which delivery has been decided, it calls the SRPI to give it a list of SRPI components able to forward the missing request. Another problem arises when EDEN is unable to decide a single delivery order for a request (or a set of requests) because there is no majority of propositions for the request (or for the set). In such a case, the SRPI components which have proposed will be asked to forward the request (or the set) to the SRPI components which have not proposed. Therefore, the SRPI components must put each new incoming request in a log before proposing its identifier to the EDEN framework.

Stabilizing When all SRPI components have proposed a same set of request identifiers, it is no more necessary for the SPRI components to keep a log of the corresponding requests. This situation is detected by the EDEN framework which calls a specific operation of each SRPI component to ask them to remove requests from the log.

5 Lessons Learned

We were confrontated with some difficulties in implementing our prototype of a FT-CORBA service solely based on the use of portable interceptors to interact with the EDEN framework. In the following paragraphs, we enumerate some of the most important problems and we discuss about the advantages and inconvenients of alternative solutions.

1. Redirecting client invocation. Due to known limitations of client portable interceptors [1] (impossibility to generate a reply or to filter among multiple replies), it's not possible to build a FT-CORBA client without the help of an intermediary object.

   That is the reason why we decided to redirect to a gateway a client invocation destinated to an object group. The gateway, which uses the CORBA dynamic interfaces, DSI and DII, behave like a server for the client and like a client for the server object group.

2. Generating object group reference. Creating a group reference is an operation driven by the replication manager. When it is being asked to create a server object group by an application, the replication manager needs (1) to create the replicas by calling the factory operation `create_object` and (2) to generate a reference identifying the group. Unfortunately, since the replication manager is a regular CORBA object without any access to the internal ORB interfaces and functionalities, it has very few means for manipulating references. The only way to act, at a service level, on the creation of a reference is to use IOR interceptors (one of the three categories of portable interceptors). IORInterceptors only enable a server to add components. Thus, using IORInterceptors is insufficient for replication manager because: (1) IORInterceptors do not provide for adding profiles, only components; (2) IORInterceptors are not invoked for already existing references, only for newly created ones; and (3) even for newly created references, it is underspecified when IORInterceptors are invoked.

3. Language Mapping. We implemented the server part of our architecture in C++. In the C++ mapping, the interceptor is allowed to access all the attributes of the `RequestInfo` structure. This way, we could get the arguments attribute and log a request such that the SRPI component be able to forward it to others when necessary. Unfortunately, with the Java portable bindings, the arguments attribute is not available. Consequently, we could not port the server part of our solution using the Java language.

4. Thread Model Dependency. The strategy we used to re-order CORBA requests is strongly related to the concurrency model selected at the server-side and to the way portable interceptors are implemented. Using a "thread-per-request" model at the server-side and

considering that, in ORBacus, interceptors are implemented in a manner such that the thread executing the interceptor code also executes the user server code, we managed to implement a scheduling policy that re-order requests (based on the EDEN decisions). Any modification to the implementation of portable interceptors in the ORB (still compliant to the specification) could lead us to reconsider our implementation. Consequently, even if we have used only portable mechanims (like portable interceptors) to implement requests re-ordering, we need to carefully examine how interceptors are implemented before to contemplate migrating to a new ORB.

6 Conclusion

Our goal was to prototype a FT-CORBA service by using the most portable features of CORBA to interact with the EDEN group communication framework. We can summarize the results of this exercice by saying that client portable interceptors are useful tools to redirect client invocation and that server portable interceptors are also useful but not sufficient to provide all required functionnality at the server side.

We are now investigating the design of a new architecture in which the EDEN framework is still cleanly interfaced with an ORB but in which interfaces could be exposed to application programmer. Of course, in that way, we loose portability but one of our goals is now to bring to developpers of fault-tolerant CORBA application the benefits of the modularity of the EDEN framework. For instance, in our prototype, we considered only infrastructure controlled consistency. Application controlled consistency is another possibility allowed by the FT-CORBA specification. In that case, it could be interesting to provide to the application developper high-level services like consensus to help it to customize specific consistency protocols.

Acknowledgements

We would like to thank Michel Hurfin and Roman Vitenberg for helping with the design of the interface between CORBA and EDEN.

References

[1] R. Baldoni, C. Marchetti, and L. Verde. ”CORBA Request Portable Interceptors: Analysis and Applications”. In *Proceedings of the 3nd International Symposium on Distributed Objects and Applications (DOA 2001)*, pages 208–217, sep 2001.

[2] R. Friedman and E. Hadad. A Group Object Adaptor-Based Approach to CORBA Fault-Tolerance. In *IEEE Distributed Systems Online*, volume 2. Special Issue on Middleware 2001, 2001.

[3] P. Narasimhan, L.E. Moser, and P.M. Melliar-Smith. Enforcing Determinism for the Consistent Replication of Multithreaded CORBA Applications. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, pages 263–273, sep 1999.

[4] P. Narasimhan, L.E. Moser, and P.M. Melliar-Smith. The Interception Approach to Reliable Distributed CORBA Objects. In *Proceedings of the third USENIX Conference on Object-Oriented Technologies and Systems* , jun 1999.

[5] OMG. Fault Tolerant CORBA Specification. V1.0. ptc/00-04-04.

[6] OMG. Portable Interceptors. ptc/01-03-04.

[7] N. Wang, K. Parameswaran, and D.C. Schmidt. The Design and Performance of Meta-Programming Mechanisms for Object-Request Broker Middleware. In *Proceedings of the 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS 2001)*, feb 2001.

[8] T. Chandra, S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(1):225–267, 1996.

[9] M. Hurfin, R. Macêdo, M. Raynal, F. Tronel, A General Framework to Solve Agreement Problems *Proc. 18th IEEE Symp. on Reliable Distributed Systems - SRDS*, pp. 56-65, 1999.

[10] R. Guerraoui, A. Schiper, The Generic Consensus Service. *IEEE Transactions on Software Engineering*, Vol. 27, No.1, pp. 29-41, January/2001.

[11] F. Greve, M. Hurfin, M. Raynal, F. Tronel, Primary Component Asynchronous Group Membership as an Instance of a Generic Agreement Framework. *ISADS'2001: 5th International Symposium on Autonomous Decentralized Systems*, pp 93-100, March 2001.

[12] F. Brasileiro, F. Greve, M. Hurfin, J-P Le-Narzul, F. Tronel, EVA: an Event Based Framework for Developing Specialised Communication Protocols. In *NCA'2001 (IEEE International Symposium on Network Computing and Application)*, october 2001.

[13] D. Powell, Group Communication, *Communications of the ACM*, Guest Editor, vol. 39, num. 4, pp. 50–53, april 1996.