The following paper was originally published in the
Proceedings of the Third USENIX Conference on Object-Oriented Technologies and Systems
Portland, Oregon, June 1997

The Interception Approach to Reliable Distributed CORBA Objects

P. Narasimhan, L. E. Moser, P. M. Melliar-Smith
Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

The Interception Approach to Reliable Distributed CORBA Objects *

P. Narasimhan, L. E. Moser, P. M. Melliar-Smith
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106
priya@alpha.ece.ucsb.edu, moser@ece.ucsb.edu, pmms@ece.ucsb.edu

Abstract

The Eternal system is a CORBA 2.0-compliant system that enhances the CORBA standard with replication and thus fault tolerance. The novel interception approach implemented in the Eternal system involves capturing IIOP-specific system calls made by the ORB, and subsequently mapping these calls onto a reliable multicast group communication system. The motivation for the use of this approach is that fault tolerance is transparent to the application objects, as well as to the ORB, and that any commercial ORB can be used with no internal modification. The interception approach exploits the performance of the underlying multicast group communication system to provide good performance.

1 Introduction

The incorporation of the object-oriented paradigm into the distributed computing model has resulted in the development of distributed object applications. Such applications must be portable, and the objects of the application must be able to interoperate when distributed across heterogeneous platforms with diverse hardware and software. The need for a standard that provides these features has led to the development of the Common Object Request Broker Architecture (CORBA).

While the CORBA standard provides for interoperability, language transparency, location transparency and portability, it does not address the issue of fault tolerance. Since there is an increasing need for reliable distributed object applications, current research is focusing on adding fault tolerance to CORBA.

* Research supported in part by DARPA grant N00174-95-K-0083 and by Sun Microsystems and Rockwell International Science Center through the State of California MICRO Program grants 96-051 and 96-052.

2 CORBA and IIOP

CORBA is a standard for communications middleware that defines interfaces to distributed objects and that provides mechanisms for communicating operations to objects by means of messages. The central idea of CORBA is the Object Request Broker (ORB), which mediates communication between client and server objects. All of the requests to, and responses from, the distributed objects are passed through the ORB.

To facilitate the interworking of commercial ORBs developed by different vendors, the CORBA 2.0 standard defines the Internet Inter-ORB Protocol (IIOP). IIOP allows objects operating over heterogeneous IIOP-compliant ORBs to interact with each other, irrespective of the internal structure of the ORBs or of any vendor-specific mechanisms. IIOP has a simple and generic interface that is designed to facilitate communication between hererogeneous ORBs. The IIOP-specific system calls invoked by the ORB are intended for the underlying TCP/IP layer.

3 The Eternal System

The Eternal system is a CORBA 2.0-compliant system that enhances the CORBA standard with fault-tolerance capabilities. Eternal exploits the facilities of an underlying multicast group communication system, in our case Totem, to provide CORBA-based applications with fault tolerance. In addition to providing reliable totally ordered multicasting of messages of the ORB, Totem provides mechanisms to deal with membership changes that occur when processors or processes fail, or the network partitions.

The Eternal system interfaces with the process group layer of Totem. The process group layer provides a simple set of group communication primitives and hides the implementation details of the underlying Totem protocols. Any multicast group communication system with an interface, membership services and guarantees similar to Totem, can alternatively be used.

4 Approaches to Fault Tolerance

Initial efforts to enhance CORBA with fault tolerance have taken an integration approach, with the reliability mechanisms incorporated into the ORB itself. With the advent of Object Services in the CORBA standard, other research efforts have taken a service approach, with the provision of a reliable object group service as part of the Object Services. To achieve the best of both of these previous approaches, we have adopted a novel ‘‘interception’’ approach.

These three different approaches are discussed briefly below and are illustrated in Figure 1. In all of these approaches, replication is employed to provide fault tolerance. The replicas of an object are considered to be members of an object group, where all of the replicas in the group have the same state. Requests can be conveyed to all of the replicas of an object by addressing the object group as a whole.

4.1 The Integration Approach

The integration approach [4], as implemented in the Electra ORB, as well as in Orbix+Isis, involves layering the ORB over a reliable ordered multicast group communication system. To enable the ORB to communicate its messages over the underlying system, adaptor objects are interpositioned between the reliable multicast system and the ORB. The mechanisms for the replication of objects and for the consistency of the replicas are embedded within the ORB, thus requiring internal modification of the ORB. The advantage of this approach is that it ensures transparency of the fault tolerance to the application objects since all of the necessary mechanisms are incorporated into the ORB itself. The application objects simply use the ORB as a communication path for their requests and responses.

4.2 The Service Approach

The service approach, as implemented in the OpenDREAMS project [2], involves providing an Object Group Service as part of the suite of Object Services that are defined by CORBA. The application objects convey their invocations and responses, via the Dynamic Invocation Interface (DII) and the Dynamic Skeleton Interface (DSI), to their associated OGS objects, which then coordinate with each other to perform the operation on the replicas of the object and to return the results appropriately. The advantage of this approach is that it is wholly compliant with the CORBA standard and requires no proprietary mechanisms. However, the fault tolerance is now visible to the application objects since the application objects must be aware of the existence of the OGS objects in order to utilize their services.

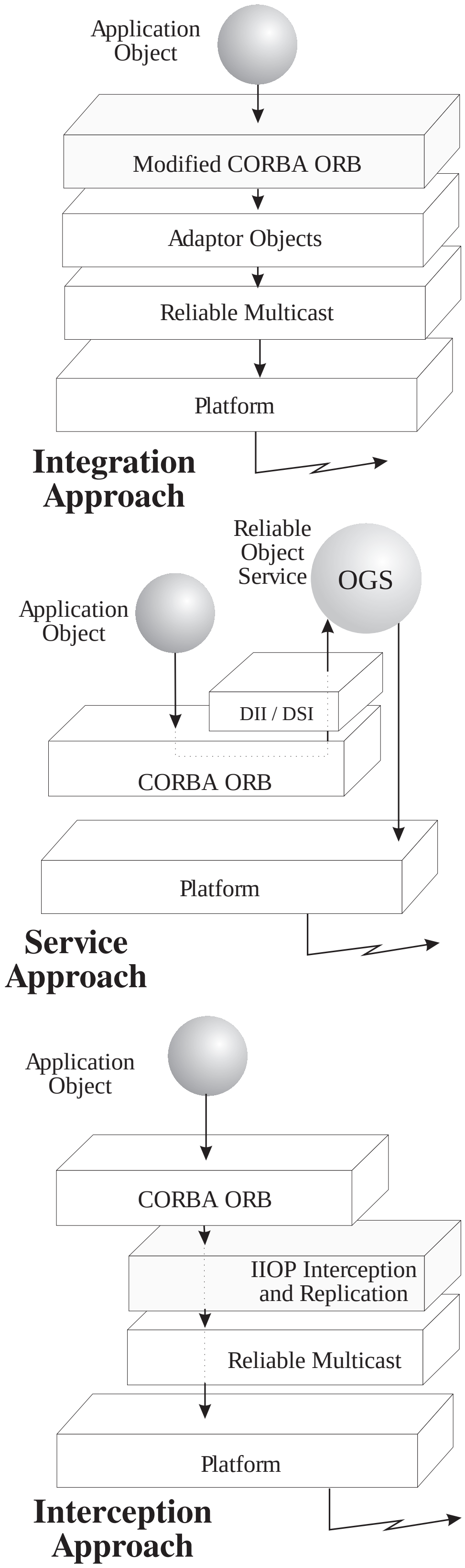

Figure 1: Different approaches to reliable CORBA.

4.3 The Interception Approach

The interception approach, as implemented in the Eternal system [7], involves capturing the system calls of the objects hosted by the ORB. The intercepted calls, which were originally directed by the ORB to TCP/IP, are now mapped onto a reliable ordered multicast group communication system. The advantages of this approach are that neither the ORB nor the objects need ever be aware of being ''intercepted'' and, thus, the fault tolerance is not visible to the application objects. Furthermore, the internal structure of the ORB requires no modification since the mechanisms that provide reliability are external to the ORB.

5 The Interception Approach in Eternal

5.1 "Catching" IIOP System Calls

Every CORBA object, on its creation, is associated with a unique Unix process identifier pid. Using user-level extensions [1] to the operating system, the system calls of the object can be traced using the file $/proc/pid$, which is a part of the $/proc$ interface in Unix. The system calls of these objects can be monitored and captured. In addition, the arguments of these intercepted system calls can be modified before the calls are allowed to proceed to the operating system.

The Eternal Interceptor ''catches'' the calls made by the ORB, via IIOP, to TCP/IP. These system calls are then mapped onto the routines of the process group interface of the Totem system, which assumes the responsibility for multicasting messages. Of interest to us are only those system calls that are invoked by the ORB for establishing connections between objects and for maintaining the interaction of objects on these connections. Thus, the Eternal Interceptor catches system calls such as *open()*, *close()*, *read()*, *write()* and *poll()*, which involve TCP/IP connections and file descriptors. This specified set of system calls finds its correspondence in the set of routines of the process group interface of the underlying Totem system.

5.2 Replication of Objects

In the Eternal system, both client and server objects can be replicated. The object group abstraction of a replicated object enables any client object in the system to address the replicas of a server object as a whole, using a unique object group identifier. The translation of the object group identifier into the individual object references of the object group members is done transparently by Eternal.

The objects of Eternal in the CORBA space are in one-to-one correspondence with processes in the Totem framework. Eternal maintains the mapping between object groups and process groups, and extends the process group membership services of Totem to object groups.

Most importantly, Eternal ensures that the states of the replicas of an object remain consistent. The reliable totally ordered multicasts of Totem guarantee that the replicas of an object ''see'' the same operations in the same order. However, in a system where replication is employed, it is possible for duplicate invocations and duplicate responses of objects to occur. These can potentially corrupt the state of an object. Eternal provides mechanisms to detect and suppress such duplicate operations.

Eternal also manages the creation of new replicas and the removal of existing ones. It also undertakes the placement and distribution of replicas and handles the degree of replication of objects.

6 Benefits of the Interception Approach

6.1 Replication Transparency

Using the interception approach, Eternal captures the calls of an object and transparently maps these calls onto an object group. Thus, a client object is only ever aware of addressing a single server object while, in fact, the request is communicated to each of the server replicas. Similarly, a server replica is only ever aware of returning its results to a single object while, in fact, the results are returned to all of the client replicas.

Replication transparency allows the application developer to write an object-oriented program for the application as if it were to run on a single machine, rather than across a distributed system. Eternal assumes the responsibility of replicating and locating the application objects, and maintaining the consistency of the replicas of the objects across the distributed system.

6.2 Use with Commercial Off-the-Shelf ORBs

The interception approach allows Eternal to ''attach'' itself transparently to any commercial off-the-shelf implementation of the CORBA 2.0 standard. Thus, the ORB itself is never aware of its calls being traced, or of the interpositioning of Eternal between the ORB and the operating system.

This implies that any application operating on any commercial ORB could take advantage of the replication and fault tolerance capabilities of Eternal without any modification to the application code or to the internal structure of the ORB.

6.3 Use of the IIOP Interface

The Internet Inter-ORB Protocol (IIOP) is supported by complete implementations of the CORBA 2.0 standard. It has a simple and generic interface, which is designed to facilitate communication between heterogeneous ORBs.

Eternal captures the calls of the IIOP interface and maps them to Totem at the client, and receives the Totem multicast messages and maps them to IIOP at the server. Thus, it is possible for the client and the server objects to be hosted on entirely different ORBs, provided that these ORBs are equipped with IIOP. Thus, the replicas that constitute an object group could, in fact, be objects implemented in different languages and running over different ORBs. The only stipulation is that these objects be able to communicate over IIOP. Fortunately, an increasing number of vendors now supply IIOP as their native protocol.

6.4 Performance

The Eternal system is currently under development, using various implementations of the CORBA 2.0 standard that are commercially available, including the CORBA-compliant Inter-Language Unification (ILU) [3] from the Xerox Palo Alto Research Center.

A typical application using a single server object and a single client object over ILU without Eternal involves 910 object invocations per second. With three-way replication of the same client and server objects over ILU with Eternal, preliminary measurements yield results of 670 object invocations per second. These measurements indicate that the overhead associated with interception and multicasting is not unreasonable for replicating objects, particularly since the code of the ORB and the operating system are unmodified. With further optimization of the code of Eternal, we anticipate even better performance.

Since replication of objects requires interaction between groups of replicas, some sort of underlying multicast group communication is required. The use of a reliable totally ordered multicast group communication system simplifies the ordering of operations at the replicas, and yields better performance than multiple point-to-point TCP/IP connections between each pair of interacting replicas. The high performance of the underlying Totem multicast group communication system is exploited by Eternal to obtain good performance.

7 Conclusion

Eternal enhances the CORBA standard by allowing application objects to be replicated and distributed on different machines across the system, while maintaining consistency of the replicas of the objects. Replication of objects across the distributed system provides tolerance to a variety of hardware and software faults. It also allows hardware and software components to be replaced while the system is live so that the application can continue to operate without interruption of service.

References

[1] A. D. Alexandrov, M. Ibel, K. E. Schauser and C. J. Scheiman, ‘‘Extending the operating system at the user level: The Ufo global file system,’’ *Proceedings of the USENIX 1997 Annual Technical Conference*, Anaheim, CA (January 1997), pp. 77-90.

[2] P. Felber, B. Garbinato and R. Guerraoui ‘‘Designing a CORBA group communication service,’’ *Proceedings of the 15th IEEE Symposium on Reliable Distributed Systems*, Niagara on the Lake, Canada (October 1996), pp. 150-159.

[3] B. Janssen, D. Severson and M. Spreitzer, ILU 1.8 Reference Manual, Xerox Corporation (May 1995), ftp://ftp.parc.xerox.com/pub/ilu/ilu.html.

[4] S. Landis and S. Maffeis, ‘‘Building reliable distributed systems with CORBA,’’ *Theory and Practice of Object Systems*, John Wiley & Sons Publishers, New York (1997).

[5] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia and C. A. Lingley-Papadopoulos, ‘‘Totem: A fault-tolerant multicast group communication system,’’ *Communications of the ACM*, vol. 39, no. 4 (April 1996), pp. 54-63.

[6] Object Management Group, *The Common Object Request Broker: Architecture and Specification* (1995), Revision 2.0.

[7] P. Narasimhan, L. E. Moser and P. M. Melliar-Smith, ‘‘Exploiting the Internet Inter-ORB protocol interface to provide CORBA with fault tolerance,’’ *Proceedings of the 3rd Conference on Object-Oriented Technologies and Systems*, Portland, OR (June 1997) (this volume).