AN OVERVIEW OF THE AQUA GATEWAY [1]

Mouna Seri, Tod Courtney, Michel Cukier, and William H. Sanders

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory and
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
{seri, tod, cukier, whs}@crhc.uiuc.edu

1. INTRODUCTION

The goal of the AQuA architecture is to provide adaptive fault tolerance to distributed applications by using commercial off-the-shelf hardware and operating systems. The AQuA architecture allows application programmers to request desired levels of dependability during applications' runtimes. It also provides adaptive fault tolerance. In distributed systems, resources change dynamically, and different types of faults can occur anywhere and anytime. AQuA is designed to provide dependability for CORBA applications. It provides fault-tolerance mechanisms to ensure that a CORBA client can obtain reliable services, even if the CORBA server object that provides the desired services suffers from crash failures and value faults.

2. THE AQUA ARCHITECTURE

Figure 1 shows the different components of the AQuA architecture in one particular configuration.

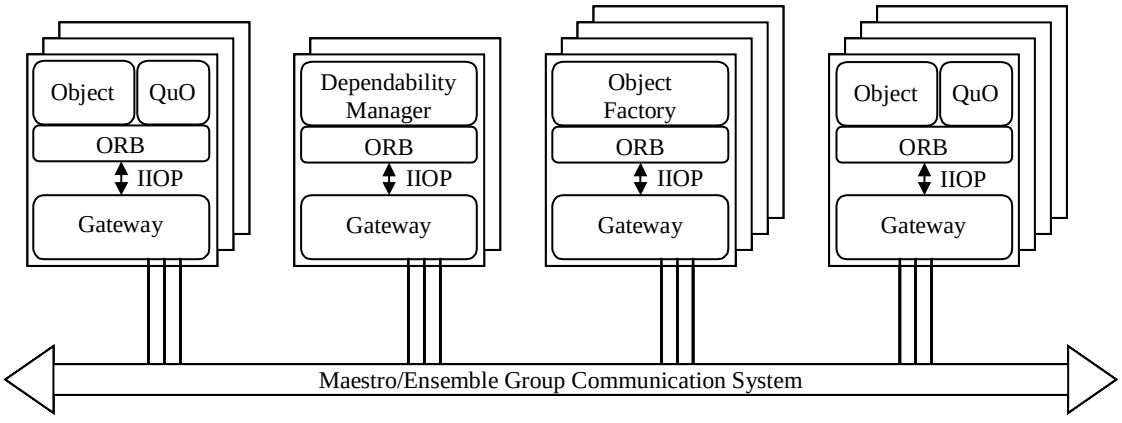

Figure 1. AQuA Architecture

[1] This research has been supported by DARPA contract F30602-98-C-0187.

In AQuA, fault tolerance is achieved through the replication of objects. All replicas of an object form a group. Messages communicated among different objects are sent through groups. A group communication system (Maestro/Ensemble [Hay98, Vay98]), as shown at the bottom of Figure 1, is used to provide reliable message multicast and totally ordered message delivery among object replicas. In addition, the group communication system provides group membership protocols to detect process crash failures and to maintain replica consistencies by transferring the state for each new group member.

In order to provide a way for an application to specify its dependability requirements, Quality Objects (QuO) [Loy98, Zin97] can be used. They allow distributed applications to process and invoke dependability requests, and to receive information regarding the level of dependability that is being provided by the current system.

In AQuA, Proteus provides adaptive fault tolerance. Proteus consists of a dependability manager, a set of object factories, and gateway handlers. The dependability manager determines a system configuration based on reports of faults and desires of application objects. An object factory that resides on each host is used to create and kill objects, as well as to provide load and other information about the host to the dependability manager.

Communication among all architecture components (i.e., applications, the QuO runtime, object factories, and dependability managers) is done using gateways, which translate CORBA object invocations into messages that are transmitted via Maestro/Ensemble. Furthermore, the handlers in the gateways implement multiple replication schemes and communication mechanisms.

The AQuA architecture is based on several groups (e.g., replication groups and connection groups). A replication group contains a set of replicas (i.e., a replica of an AQuA object that consists of an application object, the QuO runtime, and a gateway). A connection group contains two replication groups whose members want to communicate with each other. See [Cuk98] for more details on the group structure used by AQuA.

3. THE AQUA GATEWAY

The goal of this paper is to detail the component in AQuA that uses ACE/TAO: the AQuA gateway. The gateway is composed of replication handlers, handler and strategy factories, a naming service, and a DII request processor to deliver requests to the application.

The gateway is started by a simple TAO process, aquagw, whose sole purpose is to open a service configurator file and load all the services listed in the file. All the factories and the naming service are implemented as dynamic link libraries loaded as CORBA services through the ACE service configurator. The different components of the AQuA gateway are shown in Figure 2.

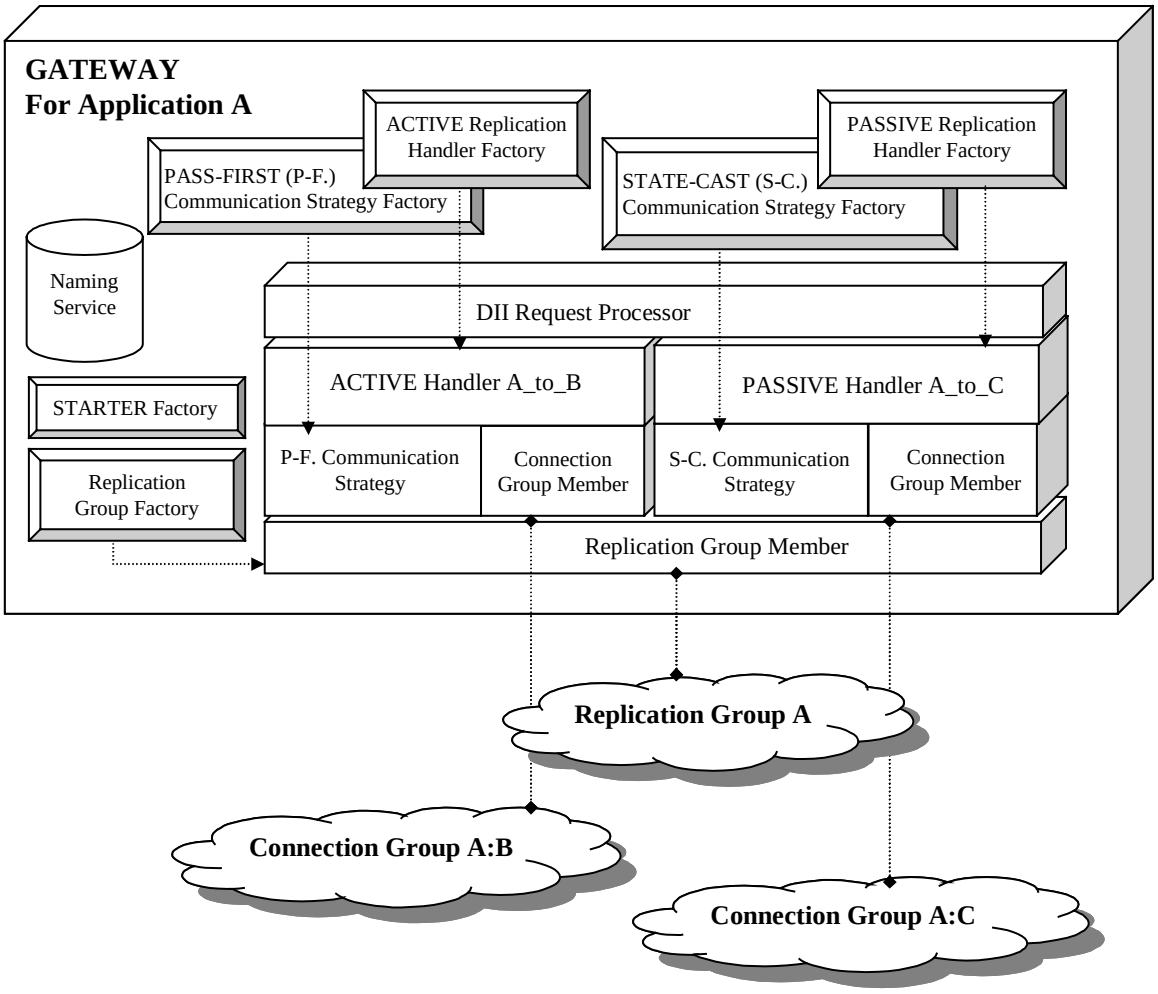

Figure 2. AQuA Gateway Architecture Shown for an Application A Communicating with Applications B and C

The replication handler is responsible for sending and receiving applications' invocations to and from a remote object through the group communication system. There is one handler for each remote object with which the local application is communicating. The handler, implemented as a CORBA DSI servant, acts transparently underneath the application and impersonates the remote CORBA object with which the application is communicating. Therefore, the application has no knowledge of the group communication system.

Each handler in the replication group is responsible for marshalling gateway messages into Maestro messages and then casting the messages to the other group members at the communication strategy's request. The receiving replication handler unmarshalls Maestro messages received through Maestro callbacks, converts them into gateway messages, and transmits them to the communication strategy for processing. In addition, the replication group members execute the Maestro state transfer from old replication group members to new ones by collecting and relaying the application and gateway state.

The handler's communication strategy processes the messages received from the application and from Maestro. The strategy is the handler's centerpiece; it makes the decisions on how to handle every message received either from the application or from Maestro, decides where to cast/send the message in either the replication group or connection group, and decides whether or not the message should be buffered or removed from the buffers.

Each handler contains a reference to the DII Request Processor, which is a service object used for delivering messages to the application.

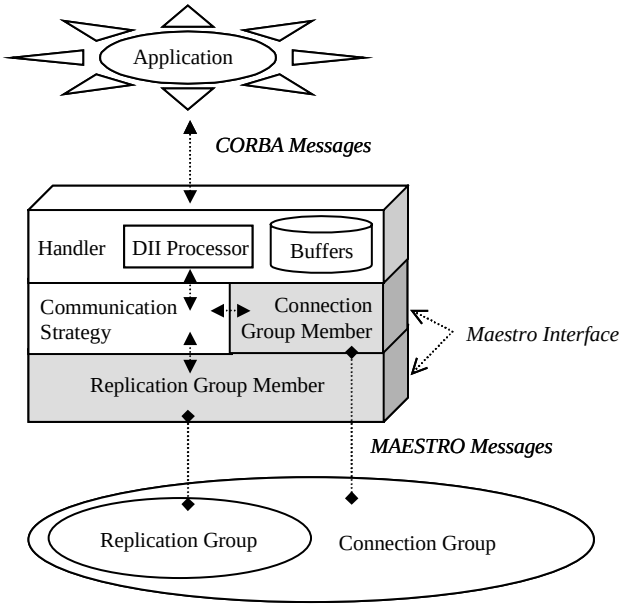

Figure 3. Handler Architecture

The naming service is unique to each gateway, and provides hash-table-based transient storage for name-to-object bindings in a naming context.

Handler and strategy factories construct the handlers and their communication strategies at initialization time. The factories register the handlers in the gateway's naming service under the handlers' name, which is provided as a command line argument in the service configurator file.

Finally, the starter factory is a service available for starting processes automatically from within the application program or the service configurator file. It is used at initialization time to start the application process after loading the naming service and before creating the handlers.

4. CONCLUSIONS

This short paper gave a brief overview of AQuA, which is an architecture that provides adaptive fault tolerance for CORBA objects. Our emphasis was on one key component of this architecture, the AQuA gateway that uses ACE/TAO extensively.

5. REFERENCES

[Cuk98] M. Cukier, J. Ren, C. Sabnis, D. Henke, J. Pistole, W. H. Sanders, D. E. Bakken, M. E. Berman, D. A. Karr, and R. E. Schantz, "AQuA: An Adaptive Architecture that Provides Dependable Distributed Objects," *Proc. of the 17th IEEE Symposium on Reliable Distributed Systems*, pp. 245-253, West Lafayette, IN, USA, October 1998.

[Hay98] M. G. Hayden, *The Ensemble System*, Ph.D. thesis, Cornell University, 1998.

[Loy98] J. P. Loyall, R. E. Schantz, J. A. Zinky, and D. E. Bakken, "Specifying and Measuring Quality of Service in Distributed Object Systems," *Proc. of the First International Symposium on Object-oriented Real-time Distributed Computing (ISORC '98)*, pp. 43-52, Kyoto, Japan, April 1998.

[Vay98] A. Vaysburd and K. P. Birman, "The Maestro Approach to Building Reliable Interoperable Distributed Applications with Multiple Execution Styles," *Theory and Practice of Object Systems*, vol. 4, no. 2, pp. 73-80, 1998.

[Zin97] J. A. Zinky, D. E. Bakken, and R. E. Schantz, "Architectural Support for Quality of Service for CORBA Objects," *Theory and Practice of Object Systems*, vol. 3, no. 1, pp. 55-73, April 1997.