An Infrastructure for Adaptive Fault Tolerance on FT-CORBA[Ψ]

Lau Cheuk Lung[1], Fabio Favarim[2], Giuliana Teixeira Santos[1], Miguel Correia[3]
[1]*Graduate Program in Applied Computer Science- PPGIA*
Pontifical Catholic University of Paraná – PUCPR - Curitiba - Brazil
[2]*DAS/UFSC – Department of System Automation – Federal University of Santa Catarina – UFSC*
Campus Universitário, Caixa Postal 476 – CEP 88040-900 – Florianópolis – SC – Brazil
[3]*LASIGE, Faculdade de Ciências da Universidade de Lisboa*
Bloco C6, Campo Grande, 1749-016 Lisboa – Portugal
e-mail: lau@ppgia.pucpr.br, fabio@das.ufsc.br, gtsantos@hsbc.com.br, mpc@di.fc.ul.pt

Abstract

The fault tolerance provided by FT-CORBA is basically static, that is, once the fault tolerance properties of a group of replicated processes are defined, they cannot be modified in runtime. A support for dynamic reconfiguration of the replication would be highly advantageous since it would allow the implementation of mechanisms for adaptive fault tolerance, enabling FT-CORBA to adapt to the changes that can occur in the execution environment. In this paper, we propose a set of extensions to the FT-CORBA infrastructure in the form of interfaces and object service implementations, enabling it to support dynamic reconfiguration of the replication.

1. Introduction

Fault tolerance support for applications developed under the CORBA distributed object model is specified in the Fault-Tolerant CORBA (FT-CORBA) standard [19]. This specification defines a set of service interfaces for the implementation of reliable applications. Fault tolerance in distributed and heterogeneous environments is usually achieved using replication techniques. The object services that provide the basic functionalities for building fault-tolerant distributed applications in FT-CORBA are the Replication Management Service, the Fault Management Service and the Logging and Recovery Service.

Even though availability and reliability are vital requirements for critical applications, the use of adaptive techniques in fault-tolerant systems is quite recent [14][7][6]. When these techniques are applied, the resource management of the computational system is significantly improved. The main idea of the adaptive approach is to provide mechanisms to the fault-tolerant system that allow it to obtain information about the execution environment; using this information, the system can adapt itself to the environment variations. Adaptations allow the system to maintain the levels of reliability and availability using only the resources necessary for the attainment of the desired requirements. The usual manner to support adaptation is the implementation of mechanisms to support dynamic reconfiguration.

The fault tolerance provided by FT-CORBA is basically static, i.e., the fault tolerance properties cannot be modified in runtime. Moreover, the system cannot be reconfigured to change the replication technique used. Support for dynamic reconfiguration of the replication mechanisms would be highly advantageous since it would allow the FT-CORBA infrastructure to cover these requirements. This support would allow the implementation of mechanisms for adapting the system to changes in the execution environment. These changes are related to the frequency/class of faults and the load of replicated components (for load balance).

In this paper, we propose a set of extensions, such as interfaces and service object implementations [19], to the FT-CORBA infrastructure to provide support for dynamic reconfiguration of replication. With this approach, the dynamic reconfiguration is entirely transparent to the application. Moreover, the model presents practical solutions to integrate requirements of Quality of Service (QoS) [23]. QoS must guide the choice of the replication technique. In this way, different levels of QoS can be specified to match different requirements of fault tolerance. We include in the Adaptive GroupPac mechanisms to identify the

[Ψ] This work is supported by CNPq (*Brazilian National Research Council*) and FA (*Fundação Araucária*) through processes 481523/2004-9, 506639/2004-5 and FA-6651/04.

necessity to reconfigure and to accomplish changes in the system with the objective of attending the QoS requirements without compromising aspects of performance and stability.

This paper is organized as follows: Section 2 presents an overview of OMG's FT-CORBA specifications. Section 3 presents the concepts related to adaptive fault tolerance. Section 4 shows the Adaptive FT-CORBA architecture (AFT-CORBA) implemented in the Adaptive GroupPac. Section 5 discusses issues related to the implementation. The model's performance assessment is shown in Section 6 and, lastly, Section 7 is the conclusion of the paper.

2. The FT-CORBA Specification

The FT-CORBA [20] architecture (implemented in GroupPac [2]) has three basic modules: Replication Management Service (RMS), Fault Management Service (FMS) and Logging and Recovery Service (LRS), beyond the definitions for interoperability in the CORBA architecture. Each module is composed by a set of services (Figure 1).

RMS interacts directly with the Object Group Management Service [20], acting dynamically in the join and leave of replicated objects. In the process of creation and removal of replicas, the object Generic Factory [20] is used for interacting with the Local Factory objects responsible for the creation and removal of replicas at the machines comprising the distributed system. The Property Management Service is responsible for defining the fault tolerance properties for each object group. This service defines the way in which each group is managed by the RMS. The Property Management Service defines, for instance, the replication technique implemented in a group, such as [20]: Cold Passive, Warm Passive and Active replication [22].

The Fault Management Service implements the interfaces of the Fault Monitoring (detection) and Notification services. Fault detection is carried out in three levels: server, object and process. These detectors are based on timeout mechanisms. The host detector is replicated to guarantee the continuity of service even when host failures occur. The Fault Notification Service performs the function of informing RMS of the faults detected by the detectors. With this notification, RMS keeps a consistent list of group members. FT-CORBA assumes perfect fault detectors [5].

The main objective of the Recovery and Logging Service is to register requests received by the server, keep the state of the replicas consistent, and carry out recovery procedures on faulty replicas. This service acts in the failure of the primary object and in the inclusion of a new member in a group. For example, when the primary object fails, the Recovery Service sends to the new elected primary the requests sent to the previous primary since the last checkpoint.

The communication between the object is carried out through the Object Request Broker (ORB), in some cases with the aid of a group communication system. The group communication system supplies the CORBA middleware with mechanisms for reliable group communication. These mechanisms are used to support the replication techniques, providing some guarantee of atomicity and order in the delivery of messages.

3. Adaptive Fault Tolerance

Fault tolerance in distributed systems has been typically provided using a combination of software/hardware redundancy, which in most cases has been statically configured. Considering the dynamic characteristics of distributed systems, like the current tendency to increase in scale, the fixed and established coordination of these models of redundancy can be quite inefficient. Static mechanisms of redundancy also have a high cost because the resources need to be allocated considering the most critical level of faults, i.e. the worst case.

Adaptive software allows its configuration to be modified to attend changes in its execution environment. In the context of adaptive fault tolerance, the environment changes can be, for example, changes in the communication standards, frequency of partial failures, types of faults, or new application requirements [11]. Adaptive fault tolerance is obtained with mechanisms that satisfy varied fault tolerance requirements dynamically through the efficient use of a limited and changeable amount of processing resources [14]. Adaptive software can involve the exchange of algorithms in execution time to attend the changes of the environment [6].

The main goal of adaptive fault tolerance is to allow that a system maintains its level of reliability and availability, through adaptation and reconfiguration in response to changes in its environment of execution or to changes in the reliability policies. With the use of static polices, the level of reliability supplied for the system will always be limited to the use of the pre-established redundant

resources. With a configurable reliability policy, the system can allocate dynamically the resources necessary to get the desired reliability. In this way, it is possible to optimize the resources usage and to reduce the cost without affecting the system performance.

4. Adaptive GroupPac Architecture

In the adaptive model, the main idea is that the developer can specify the quality of service requirements, defining the desired levels of availability and performance for this service. The adaptive model extends the FT-CORBA specifications (Section 2) including new IDL interfaces and mechanisms for dynamic reconfiguration of replication techniques. In the FT-CORBA architecture, presented in Figure 1, the services of Adaptation Management and QoS Management have been added. They are used to implement the necessary configurations to attend to the specified requirements. These services can also be replicated through the Replication Management Service in order to increase their availability.

In order to guarantee the system consistency during the reconfiguration phase, replicas stay in a suspended state. In this state, the Request Interceptors, implemented in GroupPac, are in charge of postponing the processing of client's requests and keeping them waiting until the system is able to execute these requests [13]. Using this approach, no requests are lost and the fault tolerance is improved. The extensions made to the FT-CORBA infrastructure are detailed bellow:

Replication Management (RM)

The original functionalities defined in the FT-CORBA specification were kept. The new functionalities included in this module allow the replication of the Adaptation Management Service and the QoS Management Service in order to increase the availability of the two services. The replication of these services is implicit in this module in order to guarantee the standards adopted by FT-CORBA specifications.

Adaptation Management (AM)

The function of Adaptation Management is to keep the system configuration in accordance with the level of QoS established for the application. Some aspects as the number of replicas and the type of replication used by the group are part of this configuration that is directly related with the state machine defined by the developer through the QoS Management Service.

In some situations, keeping the level of QoS desired for the application will require the system reconfiguration. The Adaptation Manager is responsible for performing these changes that are related with the join of new replicas, the exchange of the replication technique, definition of the primary member, change of the monitoring interval (of objects, processes and/or servers) and change of the checkpointing interval. In addition, there is the possibility of exchanging the primary replica (for passive and semi-active replication techniques [4][20]) in case the current primary is presenting a performance below the expected (load balancing). For example, the machine of the primary replica may be overloaded with other tasks or the service denied.

To run in a dynamic way, the Adaptation Manager monitors the Replication Management to detect failures that can make the system fail to attain the QoS level specified, requesting the system reconfiguration. Adaptation Management also keeps the QoS Management informed about the current system status. This information is related with the amount of replicas, the replication technique being used, the monitor/checkpoint intervals, the statistics of failure occurrence and the primary server performance (that can be supplied by clients).

Quality of Service Management (QSM)

The QoS Manager is an interface in which the developer or the administrator specifies the minimum QoS requirements desired (Figure 4). These requirements can be specified dynamically using a state machine (Figure 2). Each state represents a level of QoS with a configuration desired for the system, such as number of replicas, replication technique, fault monitoring intervals and checkpoint intervals (in the case of passive replication [4]).

The transition between two states can take many conditions into account, each one with a different priority degree. This approach guarantees that no more than one condition is satisfied at the same time. A transition between two QoS levels happens when the current QoS level does not attend the defined reliability requirements, e.g., when there is a detection of faults in replicas, faults in the Replication Manager or even the absence of faults during a time interval. Figure 2 presents a possible configuration of the state

machine with three levels QoS. Other combinations are possible with the insertion of new states. When a transition happens, a dynamic reconfiguration will be executed by the Adaptation Manager in accordance with the new QoS requirements. In case of change of primary member or replication technique, the requests sent during the transition phase will be kept waiting until the system is able to receive and send them to the replicated server.

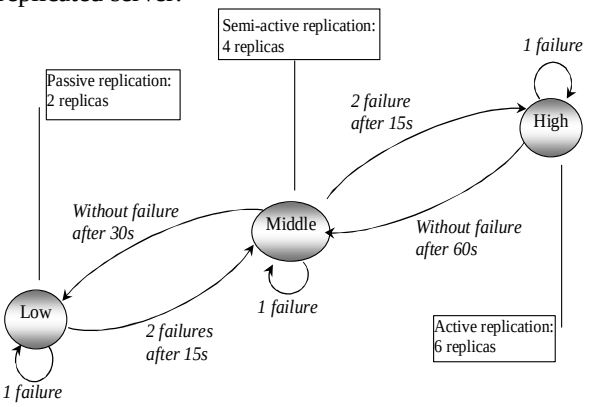

Figure 2. State machine of the QoS Manager.

Afterwards, to define the new configuration that will be implemented, the Adaptation Manager invokes the Properties Manager object service (Figure 1) to add the new fault-tolerant properties related with the replicated group. All fault tolerance properties can be modified dynamically. This characteristic differs from the FT-CORBA specification.

Finally, when the new fault tolerance properties have been modified the Object Group Manager acts in the system to implement the new replication technique. In this transitory state, the replicated server does not execute any requests. The QoS Manager is also used by the Adaptation Manager to inform the programmer about the system status, when requested.

5. GroupPac Adaptive Implementation

The implementation of the Adaptive GroupPac system was made using the Java programming language, GroupPac [16][2][3] and JacORB version 1.4, which follows the CORBA specification [20]. The class diagram represented by Figure 3 shows the classes that compose the QoS Management and Adaptation Management.

The Adaptive Manager keeps the adaptive structure of the system. This structure also contains the current number of replicas, the replication technique in use, among other information (Figure 4). This data allows the system to be reconfigured, making possible changes of some FT properties like the number of replicas, replication technique, checkpoint and monitoring intervals, membership style, consistency style, etc [20].

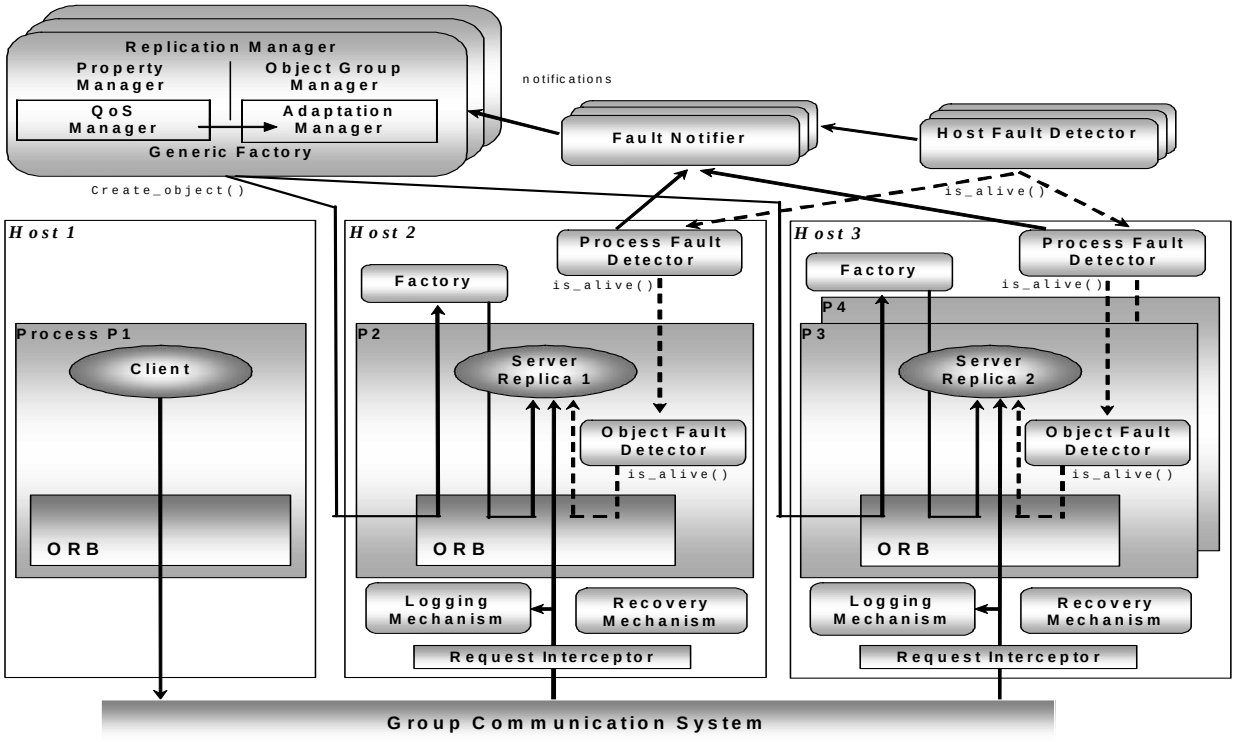

Figure 1. Adaptive Fault-Tolerant Architecture.

```
struct Transition {
   long TransitionName; long Priority;
   long Faults; long Seconds;
   string NextState;
};
typedef sequence <Transition>
                        TransitionVector;
enum ReplicationTypeQoS{none, passive,
                    semi-active, active};
struct StateQoS {
   string Name; boolean InitialLevel;
   replicationStyleQoS ReplicationStyle;
   long MinimumNumberReplicas;
   long CheckpointInterval;
   long FaultMonitoringIntervalAndTimeout;
};
typedef sequence <StateQoS> StateQoSVector;
interface AdaptiveManager {
   void set_QoS(in StateQoSVector states);
};
```

Figure 4. IDL Description of Adaptive GroupPac.

Figure 5 presents the graphical interface of the QoS Manager. This interface is used to specify the QoS system requirements, which will compose the state machine illustrated in Figure 2.

Generic Factory

The generic factory (Figure 6) of the FT-CORBA architecture is used in the adaptive GroupPac to create or remove replicas according to the configuration specified in the QoS Manager. A replicated object is created using the method create_object, from the TypeID, that identifies the object to be created by the factory. This method returns a FactoryCreationId identifier that can be used later to remove the object using the method delete_object.

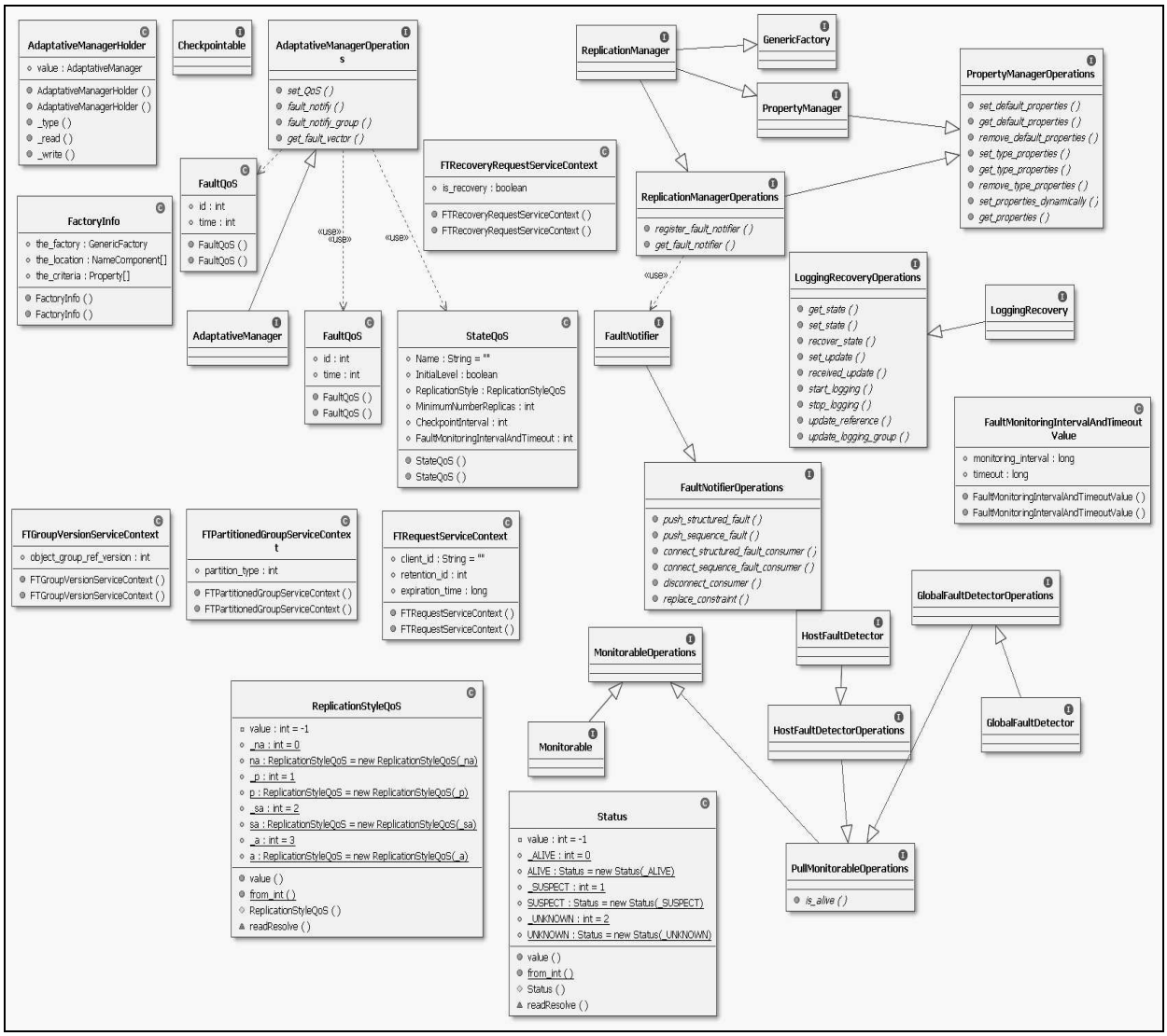

Figure 3. Class diagram of the QoS Manager and Adaptation Manager.

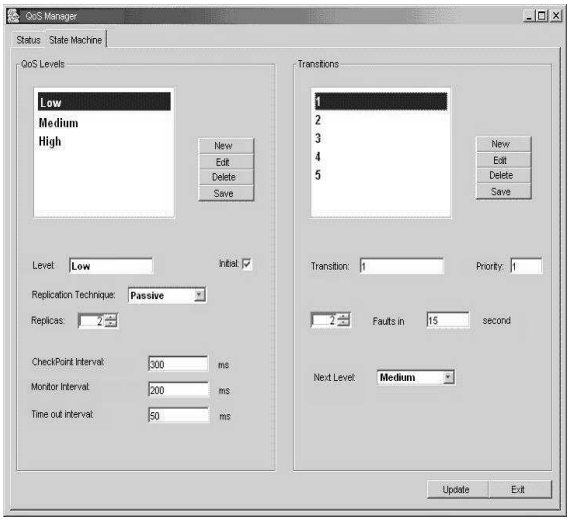

Figure 5. Graphical Interface of QoS Manager.

Request Interceptors

During the reconfiguration process, no client request can be processed, otherwise the application could become inconsistent. For example, during the change of the replication technique from passive to semi-active, a reading operation could cause the reading of incorrect data if the replicas are not synchronized. Moreover, it is necessary to assure that all client requests had been answered before the beginning of the reconfiguration, leaving the system in a consistent state [14]. We use the CORBA interceptors to make this sort of admission control. The interceptor acts as a plug-in mechanism witch can be placed in the server side as well as in the client side. There are several possible functions regarding this mechanism [20]. In our case, it is used in the client side aiming to block the execution of the client requests during the reconfiguration process.

Figure 7 presents a reconfiguration of replication technique, from passive to active, where the client sends a request M2. In the first attempt, the request M2 is sent to server 1 (primary replica), and its interceptor verifies if the system is in the reconfiguration state. In the affirmative case, an exception is returned to the client.

To guarantee the transparency in the client side (it does not need to receive the exception), the interceptor gets the exception and makes new attempts until a new primary server has been chosen and installed (when the replication technique keeps unchanged) or the replication technique has been changed. In this case, the client interceptor gets the new group reference IOGR (*Interoperable Object Group Reference* [20]) and it is informed by the Replication Manager that the technique is active replication.

```
module FT {
  interface GenericFactory {
    typedef any FactoryCreationId;
    Object create_object(in TypeId
       type_id, in Criteria the_criteria,
       out FactoryCreationId
       factory_creation_id)
       raises (NoFactory, ObjectNotCreated,
        InvalidCriteria, InvalidProperty,
        CannotMeetCriteria);
    void delete_object(in FactoryCreationId
        factory_creation_id)
        raises (ObjectNotFound);
  };
};
```

Figure 6. IDL Description of the Generic Factory.

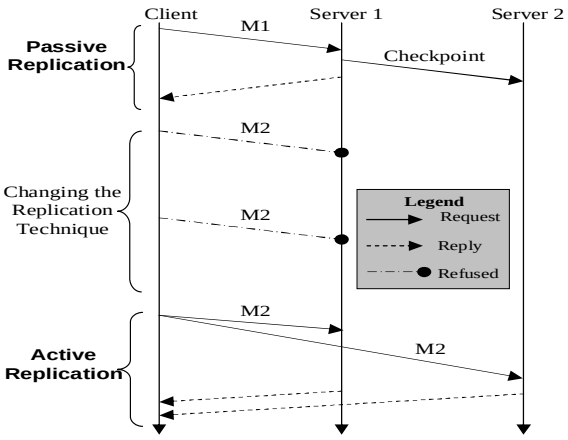

Figure 7. Interceptor actions during reconfiguration.

6. Performance Evaluation

In order to verify the performance of the implementation of the Adaptive GroupPac, some tests (obtained from the average of 1000 executions) were executed on a testbed composed by 5 Pentium IV 2.6Ghz computers with 256Mb of memory, running the Linux Mandrake 10.2 operating system and Java 1.4.2, connected by an Ethernet 100Mbps local network. All replicas were running the Generic Factory service and in one of them it was instantiated the complete system (Replication Manager, Adaptive Manager, Fault Detector, Generic Factory, Name Service and HTTP Service).

The tests were executed aiming to measure the overhead generated with the addition of the adaptive module to GroupPac. The test was executed aiming to

measure the elapsed time the system needs to recover from a fault intentionally generated. The system recovery after a fault consists of the following tasks: 1) Fault detection and its notification to the Replication Manager; 2) Depending on the replication technique, an action needs to be executed, e.g., in case of passive replication, if the fault was in the primary server, the election of a new primary server for the group will be necessary; 3) If the minimum number of requested replicas is more than the current number of replicas after the fault, it will be necessary to create a new replica for the faulty group, i.e., to make the adaptation in the system. Figure 8 illustrates the elapsed time to recover the system from a fault.

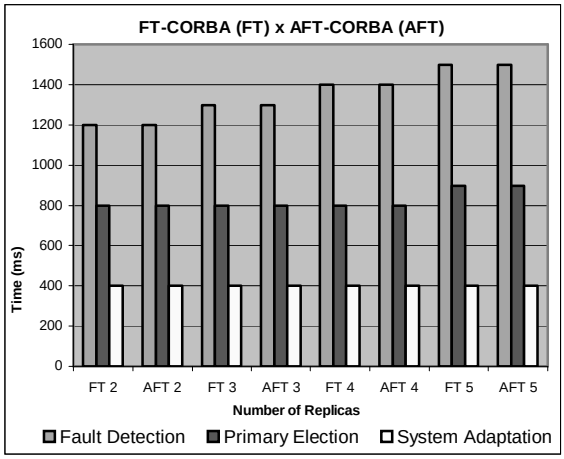

Figure 8. Performance of the Adaptive GroupPac (AFT-CORBA) and FT-CORBA.

Figure 8 shows that when the adaptive module was used, the system did not incur in additional overhead in relation to the FT-CORBA standard [20], which is a good result. This is due to the dynamic reconfiguration task, that consists of modifying the properties of the group according to the transitions between the states that define the QoS level and the creation or removing replicas in the faulty group, it is completed before the task executed by the Replication Manager in the fault treatment is finished.

The FT-CORBA already incurs in a decrease of performance in the replica creation, maintenance and removal. As AFT-CORBA uses all the infrastructure of FT-CORBA, it implies that, for example, the time to create a replica in AFT-CORBA is the same that in FT-CORBA. However, the advantage is that AFT-CORBA takes dynamic decisions that allow a better use of the system resources.

Moreover, it can be observed that in a group with only 2 replicas, the elapsed time to recover a failure in relation to the groups with more replicas (3, 4 and 5 replicas) did not increase considerably.

The reconfigurations performed by the Adaptive Manager in the groups when no faults happen in a determined time interval normally consist only in removing replicas and changing the properties of the group. These changes do not cause instability in the system execution.

We have also measured the average time to change the replication technique. In this experiment, the average time to stabilization between passive replication and semi-active replication was approximately 360 ms and the average time to stabilization between semi-active and active replication was approximately 390 ms.

7. Related Work

There exists a considerable amount of literature about the introduction of fault tolerance techniques in CORBA. These works can be classified in three approaches: integration [17][12], service [9] and interception [18][15]. These works focus basically in keeping performance and portability requirements. Conside-rations on interoperability were limited because these works had been done before the standardization of the FT-CORBA specification by the OMG.

The AQuA architecture (*Adaptive Quality of Service Availability*) [7] aims to supply adaptive fault tolerance for distributed applications. AQuA allows application developers to specify the desired levels of dependability, which are reached through the configuration of the system in accordance with the availability of resources and the faults occurred. AQuA uses QuO to specify QoS requirements at application level, and the Proteus dependability manager [21] to configure the system in response to faults and availability requirements. Ensemble [10] is also used by AQuA in order to provide group communication services. QuO and AQuA need QoS requirements to be defined at compile time, while AFT-CCM [8] allows QoS requirements to be modified at execution time. As the AQuA was developed before FT-CORBA, its architecture is not compatible with this specification.

Chameleon [1] is an adaptive infrastructure that provides different levels of availability through an architecture composed of ARMORs (Adaptive, Reconfigurable, and Mobile Objects for Reliability). Chamaleon can select different combinations of ARMORs in order to provide different availability levels. Besides providing a highly specialized fault-tolerant environment, ARMORS are also location

independent, i.e., they can execute these actions in any node in a heterogeneous network. ARMORs also allow functionalities to be introduced incrementally in the system.

8. Conclusion

In this paper, we present a set of extensions to the FT-CORBA architecture so that applications can be dynamically reconfigured according to the conditions of the execution environment. This reconfiguration is implemented using the standard mechanisms provided in CORBA [19] and FT-CORBA [20].

Although adaptive techniques are also supported in several previous works in the literature to enforce fault tolerance requirements, these works are not based totally on the FT-CORBA standard. Tests performed with a prototype have shown that the adaptation mechanisms do not cause more overhead. Aiming to provide adaptation not only at the level of crash faults, we intend to add support to other fault classes in the future.

References

[1] Bagchi, S., et al., The Chameleon Infrastructure for Adaptive, Software Implemented Fault Tolerance, *17th IEEE Symposium on Reliable Distributed Systems – SRDS'99*, pp. 261–267, West Lafayette, USA, Oct. 1998.

[2] Bessani, AN., Fraga, JS., Lung, LC., Alchieri, EAP., Active Replication in CORBA: Standards, Protocols and Implementation Framework. *6th Int. Symp. on Distributed Objects and Applications – DOA'04*, LNCS vol 3291, Larnaca, Cyprus, Oct., 2004.

[3] Borusch, D., Lung, LC., Bessani, AN., Fraga, JS., Integrating the ROMIOP and ETF Specifications for Atomic Multicast in CORBA, *7th Int. Symposium on Distributed Objects and Applications – DOA'05*, LNCS vol. 3760, pp. 680-697, Agia Napa, Cyprus, Oct., 2005.

[4] Budhiraja, N., Marzulo, K., Schneider, FB., Toueg, S., The Primary-Backup Approach, *Distributed Systems*, chapter 4, Addison Wesley, Second edition, 1993.

[5] Chandra, T., Toueg, S., Unreliable failure detectors for reliable distributed systems, *Journal of the ACM*, 43(2):225 267, Mar. 1996.

[6] Chen, WK., Hiltunen, MA., Schlichting, RD., Constructing Adaptive Software in Distributed Systems, *21st Int. Conference on Distributed Computing Systems*, pp. 635–643, Mesa, USA, Apr., 2001.

[7] Cukier, M. et al., AQuA: An Adaptive Architecture that Provides Dependable Distributed Objects, *17th IEEE Symposium on Reliable Distributed Systems*, pp. 245–253, West Lafayette, USA, Oct. 1998.

[8] Favarim, F., Adaptive Fault-Tolerant CORBA Components, Master Dissertation, Centro Tecnológico, UFSC, Florianópolis – Brazil, Mar. 2003.

[9] Felber., P., Narasimhan, P., Experiences, Strategies, and Challenges in Building Fault Tolerant CORBA Systems, IEEE Transactions on Computers, 53(5):497–511, 2004.

[10] Hayden, MG., The Ensemble System, Ph.D. thesis, Cornell University, USA, 1998.

[11] Hiltunen, M., Schlichting, R., Adaptive Distributed and Fault-Tolerant Systems, *Int. Journal of Computer Systems Science and Engineering*, 11(5):125–133, 1996.

[12] Isis Distributed Systems Inc, IONA Technologies, Ltd. Orbix+Isis Programmer's Guide, Doc D070-00, 1995.

[13] Kramer, J., Magee, JA., Model for Change Management, *IEEE Int. Workshop on Future Trends in Distri-buted Computing Systems*, pp. 296-300, Hong Kong, 1988.

[14] Kim, KH., Lawrence, T., Adaptive Fault Tolerance: Issues and Approaches, *2nd IEEE Work. on Future Trends of Distributed Computing Systems*. 38–46, Cairo, Egypt, 1990.

[15] Lung, LC., Fraga, JS., Farines, JM., Ogg, M., Ricciardi, A, CosNamingFT– A Fault-Tolerant CORBA Naming Service, *18th IEEE Symposium on Reliable Distributed Systems – SRDS'99*, Lausanne, Switzerland, 1999.

[16] Lung, LC., Padilha, R., Souza, L., Fraga, JS., Implementing the FT-CORBA Specification, Tech report, LCMI-DAS-UFSC (www.das.ufsc.br/grouppac), 2001.

[17] Maffeis, S., Run-Time Support for Object-Oriented Distributed Programming, Ph.D. Thesis, University of Zurich, Zurich, Switzerland, 1995.

[18] Moser, LE., Melliar-Smith, M., Narasimhan, P., Consistent Object Replication in the Eternal System, *Theory and Practice of Object Systems*, 4(2): 81-92, 1998.

[19] OMG, The Common Object Request Broker Architecture v3.0., OMG Document 02-06-33, 2002.

[20] OMG, Fault-Tolerant CORBA Spec. V1.0, OMG Doc: ptc/2000-04-04, Apr. 2000.

[21] Sabnis, C. et al., Proteus: A Flexible Infrastructure to Implement Adaptive Fault Tolerance in AQuA, *7th IFIP Int. Work. Conf. on Dependable Computing for Critical Applications*, pp. 137-156, S. Jose, USA, Jan. 1999.

[22] Schneider, FB., Implementing Fault-Tolerant Service Using the State Machine Approach: A Tutorial, *ACM Computing Survey*, 22(4):299-319, 1990.

[23] Zinky, JA., Bakken, DE., Schantz, RE., Architectural Support for Quality of Service for CORBA Objects, *Theory and Practice of Object Systems*, 3(1):53–73, 1997.