Adaptable Replication Scheme for Availability and Reliability in a Distributed Object-Oriented Computing Environment

Juan Carlos Leonardo　Kentaro Oda　Takaichi Yoshida
Department of Artificial Intelligence
Kyushu Institute of Technology
leonardo,ken,takaichi@mickey.ai.kyutech.ac.jp

Distributed object-oriented systems tend to evolve steadily influenced by its own requirements. Adaptability enables objects to change its respective behaviour on the fly to fulfill the current requirements of the system. As an alternative to leverage the availability and reliability characteristics, we propose an adaptable replication scheme that enables an object to adapt itself to provide a replicated service, and a replicated object to choose the replication strategy that best fits to achieve its requirements.

The adaptable replication framework encapsulates the details of replication and communication which comprises a group membership service, reliable message delivery service and message ordering service to comply with the consistency of the replication protocol. The adaptive replication scheme permits replacement of down replicas, or change of the number of replicas when partial failures occur and chooses the most adequate consistency protocol for the current configuration. Clients to the recently adapted objects for replication should take actions that enables then to obtain a thourough service of the replicated object. The adaptable replication scheme is being designed as part of the Juice system.

1 Introduction

Many replication protocols have been proposed to enhance availability, reliability and performance issues. These protocols base their replication policies on a determined computing environment and system requirements. The ROWA protocol does not consider site failures as part of its specification. The ROWAA protocol [2] [4], however, does consider site failures as part of its specification. The quorum consensus protocol [5], keeps system functionality in spite of site failures and network partitions so long as a quorum is assembled. The primary-backup protocol [3], works in a centralized way so that operations can be serialized consistently without incurring in complex cooperation issues. Other replication protocols revolve around these concerns offering different levels of availability, reliability and performance. These issues are subject to the changes in the environment. Thus, a replication protocol cannot assure high dependability at all times. Adaptive replication offers an alternative which permits to change the replication scheme during the lifetime of an object when environmental conditions has suffered drastic changes. Replicated objects with a fixed replication protocol may run into troubles whenever issues out of its scope show up.

Furthermore, singleton server objects can enhance its availability and reliability by replicating themselves. This involves the deployment of a replica group concept to encapsulate the replication details. A replicated object is accessed through single object identity throughout the whole system. Depending on the replication policy adopted, the replicated object can change the number of replicas, keep the number of replicas despite replica crashes to hold better availability and reliability, etc. The consistency protocol employ to provide correctness can be of any form than ranges from a relaxed approach to a strict approach. The object can adapt itself to adopt the consistency protocol that most suits its requirements.

Client objects can also be adapted to take advantage of replicated objects. Clients just have knowledge of the server object but not of behaviour it has currently adopted. In case of replication, it needs to know the replica group object to be able to talk to the replicated service. For that, clients must obtain this knowledge indirectly through a directory server which acknowledges clients of the server object current behaviour.

The adaptable replication scheme is designed on the Juice system. The Juice object model comprises adaptability as one of its features. It allows objects to change its behaviour on the fly by replacing some components in a modular way. A replication module can be installed to provide the adaptable replication semantics to the object.

2 The Adaptable Object Model

The Juice system [1] is based on the adaptable object model. Adaptable objects are user-defined, first-class and adapt themselves to the changing execution environment. This object model support network transparency and adaptation properties for an open distributed environment. The adaptable object model is made of five components: the encapsulation object, the executor, the communicator,the adaptation strategy and the context object (See 1).

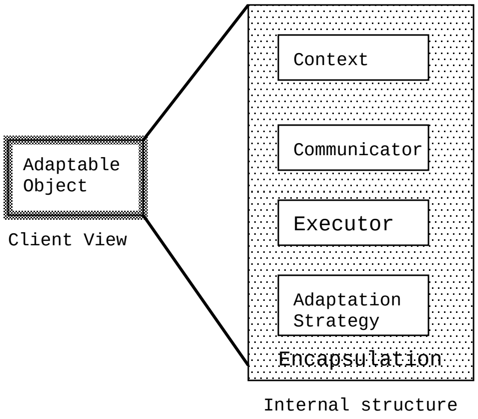

Figure 1: The Adaptable Object Model

1. Encapsulation object: It hides the internal structure of the adaptable object. The encapsulation object type is the same as the user-defined type for the problem domain.

2. The executor: It provides execution and concurrent control. It executes methods from the context object corresponding to message received from the communicator. The executor object supports concurrency control by having more than one thread of control. It permits synchronization based on mutual exclusion or condition variables. The method selector can be changed depending on the executing environment.

3. The communicator: This metaobject provides the communication infrastructure. It interprets the communication protocol, manages message reception, to say, in a message queue, sends the message in an appropriate order to the executor object. Generally, it handles both message sending and reception. The communicator can be installed modularly to provide a new communication protocol, message ordering control policy, message sending policy (unicast or multicast).

4. The adaptation strategy: provides strategies for adaptation as environment changes occur. These changes are informed in the form of events. Depending on the characteristic of the event adaptation

5. The context object: It holds the state and behaviour of the adaptable object. It merely deals with the application domain implementation. It is defined by the user.

3 The Adaptable Replication Scheme

3.1 Overview

First, the adaptable replication scheme defines the process it takes for a replicated object to distribute and synchronize messages among its members based on the current replication policy. By definition a replicated object is made of replica objects and client objects which may be located at different address spaces (See 2).

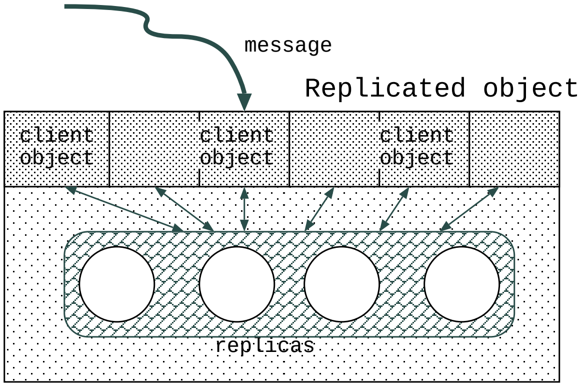

Figure 2: The replicated object framework

On the one hand, Replica objects, in general, keep the same state information and perform computations by means of a well-defined service. Issues such as data consistency, fault tolerance, performance are of concern at this level. The replication scheme expresses the different solutions to these issues. On the other hand, client objects are the ports to access replica objects' services and are allow to offer the service given by the replica objects. In the overall, this service turns to be the service provided by the replicated object. Replica objects and client objects must communicate to process messages(object-oriented environment) sent to the replicated object which is assumed to be referred to by a global object identifier. Therefore, within the replicated object, the replication scheme provides the rules for communication, the consistency protocol, and the number of replicas that abstracts replication. The replication scheme considers as the unit of communication a message. A message has it own format which is shared among all replication schemes and is analyzed and understood by any replica object and client object using the same replication scheme. An replica object receiving a message decides whether to process a message given that the message ancillary information agrees with the one currently at hand. Messages rejected by the replica service are acknowledged back to the client object. Client objects take actions based on this ackowledgement. This scheme is being supported by a communication subsystem which provides all communication services such as unicast communication and group communication. Message ordering, message reliability, message duplicates are handled at this layer.

The adaptable replication scheme also establishes the process it takes to adopt a new replication policy due to environmental conditions. For that, every replication scheme is identified uniquely. The replica set and client set must agree on the replication scheme identifier in order to communicate. Messages must carry with it the current replication scheme identifier.

The adaptable replication framework contains a input message queue, a replication component, a replication manager and a messenger (See 3). The input message queue as it names implies holds messages coming from the adaptable object itself. The messanger distributes the receiving messages to either the replication component or the replication manager. At times when the system is in a transitional stage and keeps message in a queue until further notification. The replication component holds the current replication scheme. It fetchs messages from the queue, process them adding the proper ancillary information, and sends then in the form of a request message to the communication subsystem. Further, it receives messages from the communication system, interprets the message and proceed to do its respective action. Messages that cannot be understood by the it are delegated to the replication manager. The set of replicas are allowed to change the replication scheme provided that the guarantees are not being met. Not all the members of the replicated object are aware of this kind of decisions. Therefore, such members are forced to adapt themselves to be able to establish communication again. Replication managers deal with replication scheme adaptation. No matter which replication scheme is being employed at the time, replication managers are able to communicate due to they share a common communication protocol. They are located at all members of the replicated object. They are activated whenever the replication system suffers from serious availability and reliability degradation to the extent that another replication scheme has to be adopted. Or whenever a change has been done in the system such that the current replication component fails to get a certain service.

3.2 Server Object Adaptation

Assuming that a replicated object exists server object adaptation can be observed by changes in the replication scheme and

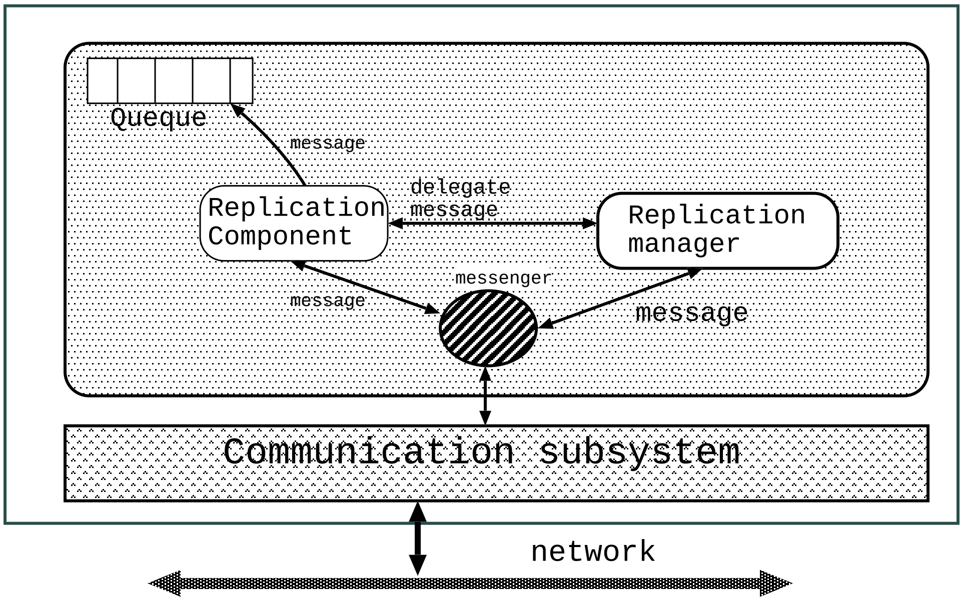

Figure 3: The adaptable replication framework

changes in the number of replicas. Replication managers, at the replica side, know each other and can communicate and cooperate to enable to replication system to evolve. In other words, any change in replication can be done by these members. Each replication manager can independently call all members for a new configuration. A manager sensing a slow down in processing can warm others managers by sending the relevant information which concerns the problem. There should a set of parameters at each replica manager that causes a spontaneous action to make changes in the system. A transitional stage is essential to prevent the messages of being process wrongly. After the change has been installed, the system can come back to normal processing. As the nature of replication protocols varies, replication components cannot present uniform characteristics. For instance, the primary replica component and the backups components differ in functionality. The primary replica handles synchronization and the backups are just mere keepers of state information. If the primary-backups approach is elected, then after the transitional stage has ended, the messenger starts sending the pending messages. Those messages are rejected and a acknowledge reply is sent back to the client objects due to the new replication scheme. In short, all pending messages are reformatted and re-sended after the client object has adopted the new replication scheme.

A replica group abstraction is introduced that simplifies the management of replicated objects. The replica group manages the number of replicas, the consistency protocol and the failure of replicas. The replica group makes use of a group membership protocol to cope with the above aspects. Every time a replica is added, or has been determined to have failed and excluded from the list of replicas, a new view of the group is created consistently. The replication protocol can, for instance, reduced the number of replicas to lessen the data consistency cost by excluding a number of replicas from the group. Or add more replicas in cases where more availability is required.

The replica group should provide a single interface for the replicated service. Clients accesses this replicated service by means of this interface. The replica group should be exported and made available to clients. A unique global logical identifier for the replica group becomes necessary.

3.3 Client Object Adaptation

Client objects should adapt to be able to get services from the replica group. The replica manager, at the client side, takes care of misunderstandings among replica components. Replica components receiving an acknowledge message can delegate the message to the replica manager. Then, the replica manager must contact the replica managers from the replica group and ask about the new requirements. During this stage the system enters in a transitional stage in which in returns the acknowledged messages back to the queue for the new replication component to process them.

Clients objects interacts with replica objects by means of a unique object reference, a kind of opaque reference, which contains the details of communication and object location, and identifies every object over the whole address space. These object references are used in the communication subsystem that needs the information to establish a connection. It is worth noting, however, that adaptability introduces an issue when the server object adopts a different approach which completely changes the communication protocol. The client and server may have an aggreement that advises the client to adapt itself to conform to the newly-selected communication protocol. A server with a single replica having a communication protocol that only understand unicast communication, due to availability and reliability degradation, can change to a multicast communication protocol to support a replication protocol. The client has to change its communication semantics and acquire a replica group reference to be able to get service done from the whole set of replicas. An smart directory service should intervene in this process by acknowledging clients that the server object has been replicated so that they start reconfiguring themselves and then interact with the replicated object. In the current approach the client can take full advantage of replication by changing its communicator to the one that provides the adaptable replication scheme.

4 Design on Juice System

The capability of the Juice system of being adaptable facilitates the integration of the adaptable replication scheme. Due to message processing is main task done within the adaptable replication scheme, a new communicator component of the adaptable object can be created which incarnates the adaptable replication scheme (See 4).

Messages are intercepted at this level and the communicator can perform additional functionality to support features. Replication as an alternative to enhance availability and reliability involves handling of messages. Messages must be recieved and executed in the same order at all replicas given that the replication protocol being employed is ROWA. They must be delivered reliably. And duplicated messages

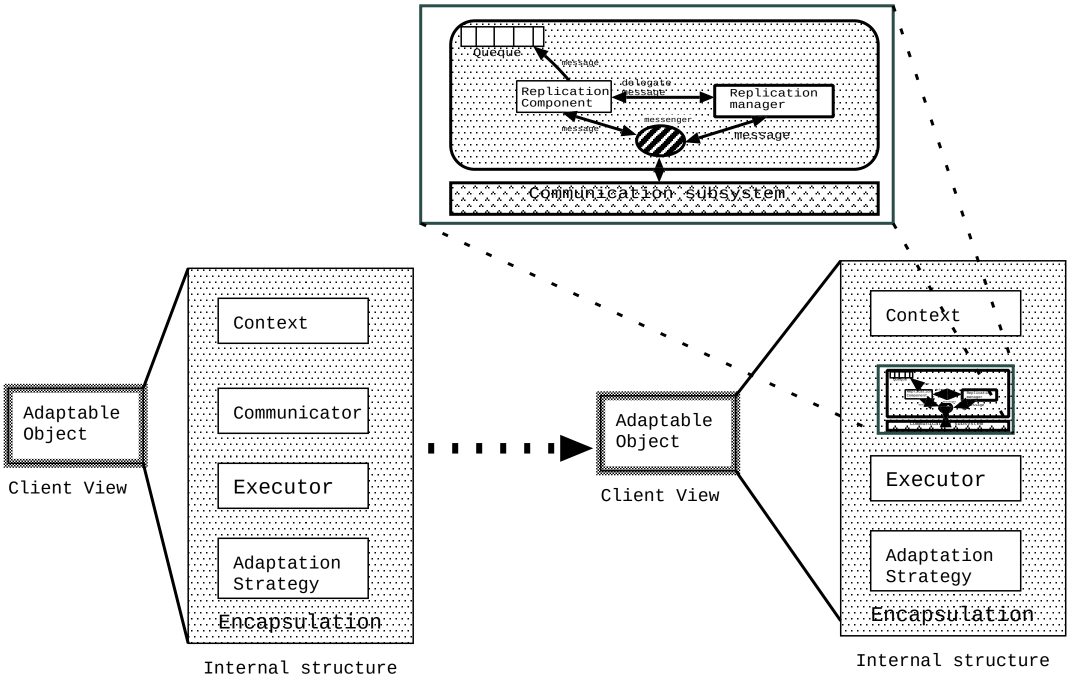

Figure 4: The adaptable object after replication adaptation

must be removed. Adaptable replication is supported at this level. The communicator with the adaptable replication scheme, hereafter replicator, deals entirely with all replication details. The replicator chores are divided into two sides: One is addressed to maintain data consistency and the other is directed to maintain group membership.

An instance of the replicator would be a communication subsystem using a token-based approach and replication policy such as active replication. A replica belongs to a replica group and receives a token that informs of the order of the previously recieved messages. It also deals with membership on the ring. Only members that belong to the logical ring are counted at the time of running the replication protocol. Failures of members must be masked transparently. The membership protocol established the rules to be applied in cases when members leave or enter the ring. In short, a new ring has to be created if one or more members fail or join the ring. For instance this approach for adaptable replication can be based on TCP connections so that the reliable delivery of point to point messages is of no concern. To provide the required availability failed replicas could be replaced for new ones spawned from any of the already-updated replicas. New replicas join the group after the new logical has formed and all replicas are up-to-date. A representative has to spawn new replicas at different processors and call for a new membership.

Global message ordering entails uniqueness of messages at all replicas. For that, every message has to be uniquely identified. A unique sequence number is attatched to each message in its header. Only the replica which has received the token can assign sequence numbers to messages following the sequence number described in the token information. The replica in charge multicasts its messages to the rest of the replicas and past the token the following replica. Due to the reliability guarantee provided by the TCP implementation, replicas has little work to do regarding message transmission reliability.

The membership protocol and the replication protocol are independent of each other so any adaptable object is free to select the replication policy that best suits its needs. The replica policy can range from a primary-backup policy, quorum consensus, mayority voting, etc. The replica group specifies the replication policy adopted. Furthermore, the replica group can resize itself to accomodate the current availability.

5 Concluding Remarks

Adaptable object model with its structure facilitates the design of adaptable replication scheme by a accomodating a communication object that specifies the replication behaviour. The adaptable replication scheme aims at providing a framework that enables adaptable objects to change the replication at runtime. The replication policy can be adopted according to the needs of each adaptable object. Adaptable objects can be enhanced in such a manner whenever changes on the environment turn out the working place hostile. An environment experiencing making the service of some objects poor available or reliable can advice the objects through events to adapt to a replicated service. The communicator with the adaptable replication scheme is part of the Juice system.

References

[1] Oda, K. and Tazuneki, S. and Yoshida, T.: The Flying Object for an Open Distributed Environment, *15th International Conference on Information Networking* (2001).

[2] Berstein, P. and Goodman, N.: An algorithm for concurrency control and recovery in replicated distributed databases, *ACM Transactions on Database Systems*, Vol. 9, No. 4, pp. 596–615 (1984).

[3] StoneBraker, M. and Neuhold E.: Concurrency control and consistency of multiple copies of data in distributed INGRES, *IEEE Trans. on Software Engineering*, Vol. 3, No. 3, pp. 188–194 (1979).

[4] Berstein, P. A. and Hadzilacos, V. and Goodman N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).

[5] Gifford, D. K.: Weighted voting for replicated data, *Proc. 7th Symp. on Operating System Principles*, pp. 150–162 (1979)