

Enhancing Availability of Cooperative Applications through Interoperable Middleware

Roberto Baldoni Carlo Marchetti

*Dipartimento di Informatica e Sistemistica
Università “La Sapienza” di Roma
Via Salaria 113, 00198, Roma, Italy*

{baldoni, marchet}@dis.uniroma1.it

1. Introduction

A *cooperative information system* (CIS) is a collection of cooperating, distributed information systems interconnected by a wide and complex communication network ([5]). CISs are characterized by high heterogeneity in terms of platforms and computing environments that make difficult the cooperation among remote entities. However, the CIS model can be applied in many “*e-contexts*”, such as *e-government*, *e-business* etc. and different proposals ([6]) are trying to address the problem of developing services for cooperative applications in such system model.

The *Unitary Network of the Public Administration* (RUPA¹) is a challenging project promoted by the Italian Government to integrate at the application-level the large set of autonomous and heterogeneous information systems of the Italian Public Administration, which in the past years followed an independent and uncoordinated design and development [1]. In RUPA, the heterogeneity of these systems has been shielded by a *cooperative architecture* that uses application gateways to standardize the access to heterogeneous resources contained in a RUPA *domain*, i.e. RUPA abstraction of an information system belonging to a single organization. Gateways are object-oriented wrappers of the domain systems providing services according to a predefined interface. Being these gateways the access point for all the services exported by a RUPA domain, their fault tolerance and high availability are key issues in order to guarantee quality of service (QoS) to RUPA clients.

Fault-tolerance and high availability can be obtained by software redundancy: a service can survive to a fault if it is provided by a set of server replicas hosted by distinct servers. If some replicas fail, the others can continue to offer the service. At this end, servers can be replicated according to one of the following replication techniques: active replication or passive replication (primary-backup approach) [3].

Due to the heterogeneity of software platforms, a CIS is a typical example of a three tier architecture. Clients interacts with application servers that, in turn interacts with backend servers such as databases. Handling software replication, in such context, is actually a new challenge as typical approaches used in two tiers replication seems does not fill well. As an example, in three tiers systems, the backend servers are autonomous entities that should not be aware of their replication paradigm.

This means that handling replication in three tiers architecture needs special platforms such as IRL (Interoperable Replication Logic) which is an interoperable middleware currently developed in our

¹ The acronym of RUPA comes from the translation in Italian of “Public Administration Unitary Network”, i.e. “Rete Unitaria della Pubblica Amministrazione” (RUPA).

department. In the rest of the paper we present how IRL functionalities can be used in the context of RUPA cooperative architecture in order to increase the availability of the application gateways.

2. The Unitary Network of the Italian Public Administration (RUPA)

The Unitary Network of the Italian Public Administration (RUPA) is the most important and challenging project undertaken by the “Autorità per l’Informatica nella Pubblica Amministrazione” (AIPA) [Authority for IT in the Public Administration]. RUPA final objective is the implementation of a “secure Intranet” interconnecting the information systems of different Italian public administrations (e.g. departments, ministries and organizations). RUPA architecture consists of three functional layers: *Transport*, *Interoperability* and *Cooperative Service*. The Cooperative Service layer includes the middleware services that enable the development and the deployment of new inter-administration cooperative applications. The distributed computing model based upon the Cooperative Service layer is called *cooperative architecture*.

A RUPA fundamental concept is the *domain*: it includes all the computing resources, networks, applications and data that belong to a *specific* administration, regardless of the technical nature of such information systems. Each domain is modeled as a single entity, regardless of its internal complexity and geographical extension. The domain functionality is made available through a *domain cooperative gateway* (DC gateway) that implements the set of services described in the export interfaces of the domain². Inter-Domain cooperation in RUPA follows the cooperative architecture model. Interfaces of the services that a domain exports can be specified in accordance with the DOC approach: first, a collection of business objects and components is specified using an appropriate *Interface Definition Language*, then implemented in DC gateways, and finally made available to other domains by deploying the gateways as object/component servers (e.g. CORBA server objects). Each domain is therefore required to define all the interfaces to be exposed and to implement them in order to make accessible the services they offer. End users and applications belonging to a domain interacts with a DC gateway by means of a local *Domain Proxy Gateway* (DP gateway) that has the knowledge of the services available in RUPA and of the protocols and interfaces required to access them. DC gateways actually become the mid-tier of a layered architecture and carry out the principal function of encapsulating existing back-office systems in object-oriented wrappers that are seen as business objects from RUPA clients point of view. A business object the object oriented provided by a system belonging to a domain. RUPA clients perceive a fault of a particular DC gateway business object (which is a wrapping of a particular domain service) as the fault of the entire domain system, even if this system is still up and ready to accept service requests. A fault of the gateway itself can result in the loss of availability of all the services exposed by the domain. Thus a DC gateway, or better all the business objects it hosts, are mission critical objects from RUPA QoS point of view.

3. Interoperable Replication Logic

The Interoperable Replication Logic³ (IRL) is a FT-CORBA compliant infrastructure of CORBA servers by using a set of CORBA objects (IRL components) which allow transparent server interactions and server

² Note that a DC Gateway can be either *centralized*, i.e. composed by one or a few hosts deployed over a LAN, or even *distributed* over a geographical area.

³ We call *replication logic* the set of protocols, mechanisms and services that have to be used in order to implement a replication technique, i.e., to offer fault-tolerance of a specific entity.

failovers to clients. As the cooperation among IRL component is carried out using standard CORBA invocations, IRL allows the deployment of CORBA server objects and IRL components over ORBs from distinct vendors (i.e., interoperability property).

IRL Architectural Overview. IRL infrastructure is made up of a set of components interposed between a client and a set of replicated CORBA servers. Each request to an object group is transparently handled by IRL, giving the illusion to clients to interact with a singleton object.

Each component is deployed in a separate process and is replicated in order not to represent a single point of failure for the entire infrastructure. Component replicas can be flexibly distributed on distinct FT-CORBA Fault Tolerance Domain⁴ hosts, allowing a (static) load balancing of IIOP connections and more effective fault tolerance. Figure 1 shows IRL components. These are classified in *host specific components* (that have to be installed on each FTD host) and *domain specific components* (one logical instance per FTD has to be installed). Each IRL component is thus replicated using a passive replication technique.

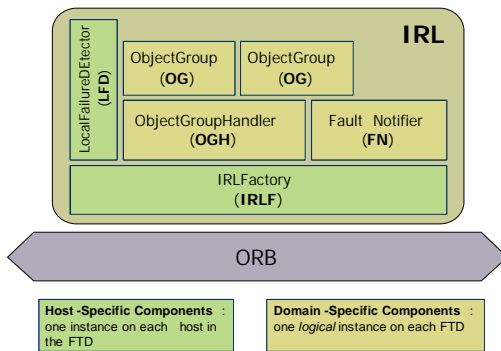
Host Specific Components:

IRL Factory (IRLF). A distinct IRLF component runs on each FTD host and exports interfaces for creating IRL components at infrastructure start-up or after a component crash.

Local Failure Detector (LFD). LFDs are FT-CORBA compliant failure detectors. A distinct LFD component is deployed on each FTD host and is responsible for detecting failures of all the monitorable CORBA objects running on its own host.

Host specific components are installed on each FTD host and their activities are related to that host (i.e., LFD monitors objects running on *its* host, IRLF creates objects on *its* host). As a consequence, host-specific components do not need to survive to their host crash.

Fig. 1. IRL Components



Domain Specific Components

Object Group (OG). An Object Group object is associated to each group of replicated application CORBA servers. OG stores the IORs of the members, the information about their availability and the properties of the group (e.g. replication style, membership style etc. [4]). OG is also responsible for guaranteeing strong replica consistency within its group.

Object Group Handler (OGH). One Object Group Handler object is associated to a FTD. This component is responsible for creating new object groups, for the management of their fault tolerance properties and for publishing their Interoperable Object Group Reference.

Fault Notifier (FN). The Fault Notifier is a FT-CORBA compliant **FaultNotifier**. It receives fault reports from LFDs and subscriptions for failure notifications by any interested client object. When FN receives a failure notification from a LFD, it forwards this notification to any interested client. In addition to this, FN implements host fault monitoring by receiving heartbeats from LFDs.

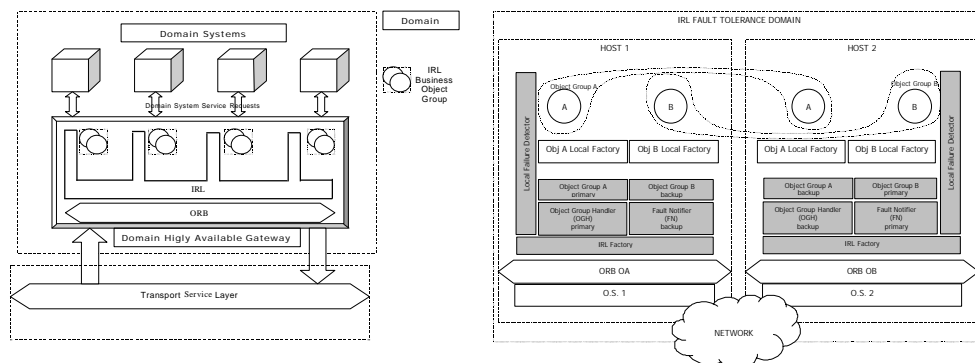
⁴ The FT-CORBA specification introduces *fault tolerance domains* (FTDs) to allow application scalability. A FTD is composed by several interconnected CORBA hosts housing members of possibly distinct object groups. Each replica of a same object group has to be deployed on a distinct host belonging to a single FTD. Distinct FTDs can be interconnected by *inter-domain gateways*. In this paper we assume that a RUPA DC gateway is a FT-CORBA FTD.

4. IRL-Based Highly-Available DC gateways

In order to preserve the autonomy of the administrations, AIPA has neither fixed a particular platform nor a specific technology on which deploy DC gateways. IRL interoperability allows the deployment of the architecture itself as well as of application objects replicas on different ORBs running on different platform. Moreover IRL can be installed without requiring any system upgrade and on top of different ORBs⁵. Deploying IRL on the domain gateways allows the replication of the business objects, making them fault tolerant and highly available. Replicating all the business object of a particular DC gateway actually means replicating the gateway and thus increasing the availability of all the domain services by decoupling the service availability from the DC gateway availability. In Figure 2.a is depicted an IRL deployment in a DC gateway.

Figure 2.b shows a possible configuration of an IRL-based highly available DC gateway housing two replicated business objects, A and B. To avoid the presence of a (*hardware*) single point of failure, business objects as well as IRL components are replicated on two hosts (Host1 and Host2). Business objects are replicated using a passive replication technique. A replicated business object is a business object *group*. Passive replication allows distributing the two primary replicas on distinct hosts, sharing the computational load between the hosts.

Fig. 2. An Highly-Available DC gateway (a) and a possible IRL deployment (b)



References

- [1] AIPA: "La Rete Unitaria della Pubblica Amministrazione". Roma, Gennaio 1996. [http://www.aipa.it/attivita\[2\]/reteunitaria\[1\]/fattib\[1\]/index.asp](http://www.aipa.it/attivita[2]/reteunitaria[1]/fattib[1]/index.asp) (in italian)
- [2] The IRL Project: <http://www.dis.uniroma1.it/~irl>
- [3] R. Guerraoui, A. Shiper, "Software-Based Replication for Fault Tolerance", in A.K. Somani, N.H. Vaidya (eds.), Special Section on Fault Tolerance, IEEE Computer, April 1997.
- [4] Object Management Group (OMG), Fault Tolerant CORBA Specification, V1.0, OMG Final Adopted Specification, OMG Document ptc/2000-04-04 ed., OMG, Framingham, MA, 1999.
- [5] M. L. Brodie, "The Cooperative Computing Initiative. A Contribution to the Middleware and Software Technologies", GTE Laboratories Technical Publication, 1998, <http://info.gte.com/pubs/PITAC3.pdf>
- [6] A.K. Elmagarmid, W.J. McIver Jr. (eds.): "Special Issue on Digital Government". IEEE Computer, vol. 34, no. 2, February 2001.

⁵ We have developed an IRL prototype running on three Java ORBs (JacORB, Orbacus and OpenORB).