Chapter 2

Policy-based Architectures

This chapter aims to present the bases on which the proposed policy system was created. In this way, it shows a short introduction about main policy system architectures and their components. The chapter has two main sections, one dedicated to the policy architecture proposed by the Working groups from the Internet Engineering Task Force (IETF) and the other section dedicated to the OMG proposal.

The IETF policy framework uses policy roles that are very important in the development of our work, mainly in the designed policy selection process and in the proposed policy conflict resolution methodology (see chapter 6). Therefore, this section shows the concept of roles shortly and the way in which we use them in our System

On the other hand, we will present the contributions that we have carried out in the environment of intelligent buildings using policies based on the CORBA (Common Object Request Broker) platform [CORBAv3].

2.1 IETF Policy System Architecture

The policy-based management networks have been the cause of wide researches in the last decade in the IETF working groups, in the Industry Consortium, which is in charge of deepen and develop all management standards about systems connected to the net, and in the DMTF (Distributed Management Task Force) working group.

The IETF Policy Framework working group created in 1996 a protocol called Common Open Policy Service (COPS) [Durham00], which originally was due to integrated services. Nowadays this protocol is used as a general-purpose policy protocol and it has different extensions such as the provisioning extension (COPS-PR) [Chan01] that is useful to support configuration management.

The IETF and DMTF working groups work together in the creation of different standards related to policies: The IETF is in charge of the architecture definition [IETF-1][IETF-2] and the DMTF defines a standard information model [CIM].

Microsoft and Cisco Systems created a common directory infrastructure able to unify the management of their products called DEN (Directory Enabled Network) [DEN] in 1998. This initiative considers an information model and a directory scheme to integrate the networks with the directory services. Thus, instead of managing individual devices, it is possible to define and to manage the rules that control the network and its resources in a central way. At the end of 1998 the DEN moved to the DMTF and together with the IETF created a common information model to describe the computers characteristics, expanding to big-scale computers networks and other devices. Such specifications form the Policy Common Information Model [Moore01], which will be explained in detail in section 2.1.4

The two IETF working groups that study policy-based networks management are:

a) The policy framework working group, which has written several drafts and RFCs [Moore01], [Moore03], [Westerinen01] to represent, manage, share and reuse all policies in a scalable way, interoperating and in an independent way from the hardware and software platform used.
b) The Resource Allocation Protocol working group (RAP), whose initial purpose was to establish a scalable policy control model for RSVP. The RAP group defined a policy framework to the admission control [Yavatkar00] consisting of management interfaces to introduce policies, repositories for their storing, decision points (PDPs) to evaluate policies and application points (PEPs) to configure the policy decisions in the network nodes.

2.1.1 Policy Framework Components

The proposed methods by the RAP working group of the IETF to manage networks based on policies were first developed in the context of integrated services, where the idea of using a policy framework to manage the admission control was proposed. This proposal is independent from any service model, so it was soon thought that it could also be used in an environment with Differentiated Services and even in other general problems.

Following, there is a description about the main components of the Framework.

Policy Management Application. It supplies the interfaces to make the network administrator specify and store the policies in a repository. It also acts as a monitor of the state in the network managed by means of policies.

Policy repository. It is a storage that uses the decision points (PDPs) to recuperate policies. The use of an access protocol is required to be able to accede. The IETF suggests the use of a Lightweight Directory Access Protocol (LDAP) [Hodges02], but other possible solutions such as other directories, databases or web servers are also available.

Policies are stored in a high level and are independent from the network devices; an information model about the devices in the network (DEN specification of Microsoft and Cisco [DEN]), users and organisations (LDAP) and applications (rule CIM of DMTF [CIM]) is also stored.

Policy Decision Points. They are the points where all decisions that must be applied on the network are created. PDPs process the policies and information about the network state to decide which policies are necessary to apply in the network. These policies are sent as configuration data to PEP or PEPs correspondents. The architecture considers that there is a PDP component and one or several PEPs that rule all physic devices.

The PDP is responsible for locating the set of rules that a PEP has to apply, recuperating them from the repository and transforming them into a format and syntax that can be understood by the device and be distributed to the PEPs. In the same way, the PDP acts as a monitor for the network state and check if all required conditions are satisfied to the policy application. If the PDP relevant policies change, then the PDP must load the repository policies again.

Policy Enforcement Points (PEP). They are the items involved in the execution of the policies that the PDP orders. Some examples of these PEPs are routers, firewalls, proxies, etc. Every PEP receives the policies in the shape of specific configuration actions and it is responsible for their establishment in the device. PEPs can also inform the PDP when any unknown situation is produced.

When a PEP receives a message that requires a policy decision, it consults a local configuration database to identify the policy items that are evaluated locally; then the PEP passes this requirement and the items evaluated to the Local Policy Decision Point (LPDP) and receives the result. Afterwards, the PEP passes all policy items and the partial results to the PDP to combine these results with the LPDP partial results and return the final policy decision to the PEP.

The IETF framework establishes COPS [Chan01] to transfer policy decisions to the PEPs and to transfer requests from PEPs to the policy server. The model is open to other mechanisms such as HTTP, FTP or SNMP.

Following figure shows the policy architecture proposed by the IETF.

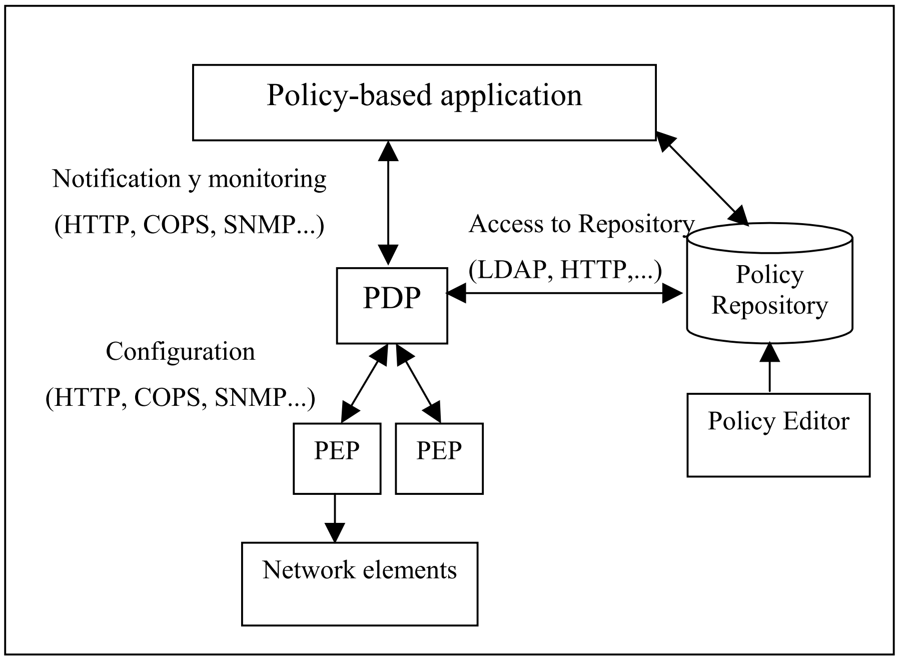

Figure 1. IETF Policy Framework

2.1.2 Policy Structure

A policy can be seen as a governing body that points the network behaviour. The IETF policy working group defines the policies as " ...an aggregation of rules, where every rule includes a set of conditions and its correspondent set of actions" [Moore01]. The policies have the shape of: "If there is (a set of conditions), then (a set of actions) can be done". The condition expression can be a simple or compound one that is related to different entities such as hosts, applications, protocols, users, etc. The action expression is a set of instructions that specifies the guaranteed or denied services or other parameters used to provide a different quality in one or more services. If the condition or conditions are true, then the action or set of correspondent actions is executed.

Conditions and actions have two main components: operating and operators. The operating components can be constant or variable and they are examined to know if the condition or the whole of conditions are true or false. Therefore, it is important to establish the value ranks that any operating can take clearly.

Policies can be simple (they are expressed in simple statements like "condition-action") or compound (to execute more elaborated functions). In the compound ones, the execution order that actions must follow can be specified. In case there is not any established order, then the actions are executed sequentially.

Policies can be implemented in different programming languages. More developed free-distribution languages are presented in section 3.2.6 The policy structure is kept independently from the language used. It is convenient to observe the following recommendations in any language used:

Preciseness. Policies must count on a detail level in order to be understood by the network items, that is to say, there must not be any ambiguity about their interpretation.

Consistency. If many policies are defined to a network item, policies must be aware between them. There must not be any ambiguity or contradiction in their interpretation, for example, if a policy assigns high priority to all accounts department packages and other policy establishes that the traffic to the human resources department must be rejected, then a router will not know whether to give high priority or reject the packages going from accounts to human resources.

Compatibility. The set of policies ready to be implemented in a device must be designed taking into account the capacity that the network nodes can support.

Once the policies have been defined, we have decided to group them depending on their aim. In this way, we have created policy roles whose functioning is explained in the following section.

2.1.3 Policy Roles

The role concept is essential in the policy framework design proposed. There are some researches referring to the policy role study, for example, [Lupu99] [Hamada98]. We have established that a role is defined by metrics (for example, in case of routing, there are three metrics: bandwidth, delay and losses) and it uses a policy selection method with representations in a hyperspace (see chapter 5) to select one or more applicable policies in a whole of entities and/or components.

A high-scale system is managed using hundreds or thousands of policies from which only a small policy sub-whole achieves the desirable behaviour in the network at a specific moment. In this sense, the policy grouping into roles is basic both in the policy selection process that a PDP has to apply and in the conflict resolution process in which several policies can be applied to the same event. Another role advantage can be seen in the implementation stage in which objects are not managed individually, but they are grouped into roles depending on their function.

There can be as many roles as necessary in a system, for example, motivation policy roles, configuration ones, installation ones, failure ones, use ones, security ones, service ones, admission control ones, etc.

Roles form a hierarchic structure in which each policy is a member of at least one role. A policy can belong to several roles. Policies are added initially to a role in the edition process (in the implementation, it happens in the object creation process) and it is necessary to specify an initial role.

Hierarchic relations among roles suppose that changes in the policies of a father role affect the objects that are in sub-domains, but if policies of a son domain change, that only affects the members of that domain and its corresponding sub-domains.

Due to the fact that a role defines a particular function of an entity, the network nodes can be associated with a role in order to achieve the behaviour according to what the policies specify for this role. In this way, the policy system proposed configures each of the network resources associated to a role. When the network behaviour changes, then the policy administrator executes an update of the policies belonging to a role and the policy framework will check that the necessary configuration updates are executed on all the resources belonging to that role.

Genetic algorithms use can make the creation of both rules and roles easier in a dynamic way within the LDAP policy directory. The use of intelligent agents for the network monitoring can also let a fast update of the knowledge base in order to allow the dynamic creation of rules and predictions of the network behaviour (traffic, failures, etc.) [Barba02].

2.1.4 Policy Core Information Model (PCIM), Policy Core Information Model extensions (PCIMe) and Policy Core LDAP Schema (PCLS)

PCIM, PCIme and PCLS are documents that present the object-oriented information model for representing policy information developed jointly in the IETF Policy Framework WG and as extensions to the Common Information Model (CIM) [CIM] activity in the Distributed Management Task Force (DMTF).

First, PCIM is briefly described [Moore01] and their extensions [Moore03]. Second, the PCIM mapping is described in a LDAP directory structure [Strassner02]. The PCIM extensions mapping in the LDAP scheme [Reyes03] is analysed in detail in chapter 4 and in appendix I.

Policy Core Information Model (PCIM) and Policy Core Information Model extensions (PCIMe)

The Policy Core Information Model and their extensions define the generic structure of a policy without taking into account its content. In this way, policies can be implemented in heterogeneous network environments.

The initial application of policies was in the ITEF, where policies were represented for the QoS management, differentiated services [Blake98] integrated services [Braden94] and for IPSec [Kent98].

PCIM model defines two hierarchies of object classes: structural classes representing policy information and control of policies, and association classes that indicate how instances of the structural classes are related to each other. PCLS and PCLSe define mappings of this information model to various concrete implementations, for example, to a directory that uses LDAPv3 as its access protocol.

PCIM classes are organised as an extensible hierarchy of classes through specialisation to define policy objects that allow application developers and the network administrator to represent different kinds of policies.

PCIM also contemplates that policies can be simple (they are expressed in simple "condition-action" statements) or compound (to execute more elaborated functions). Another important concept in the PCIM is the reusable policy, that is to say, those actions and conditions that can be associated with more than one policy. There is not any inherent difference between a condition or a specific policy action and a reusable one. However, there are differences in the way they are treated in the repository.

PCIM does not present any particular system design or any implementation and it does not define a protocol or own any specific security requirements. However, there are several documents derived from PCIM that take into account both security and more specific design considerations. PCIM is also independent from any specific repository or access protocol, even when most of researches on this aspect use a LDAP directory, for example [Salceda 02].

One of the main PCIM aims and its extensions is to create a bridge among the high-level policies that the human administrator defines and the real application commands that must be executed in the network nodes to reach the business aims planned by the administrator. Another aim consists of making the interoperability easier among different systems. That is the reason why all PCs belonging to divers systems understand the same policy semantic and own the common knowledge about how to store policies in the repository, despite the fact that every PC can interpret policies in different ways.

PCIM and its extensions are not a final scheme. There are several models derived from PCIM to solve specific management problems in determined areas, for example, QPIM proposes QoS policies, ICPM for security policies, QPLS for policies based on MPLS, etc. See figure 2.

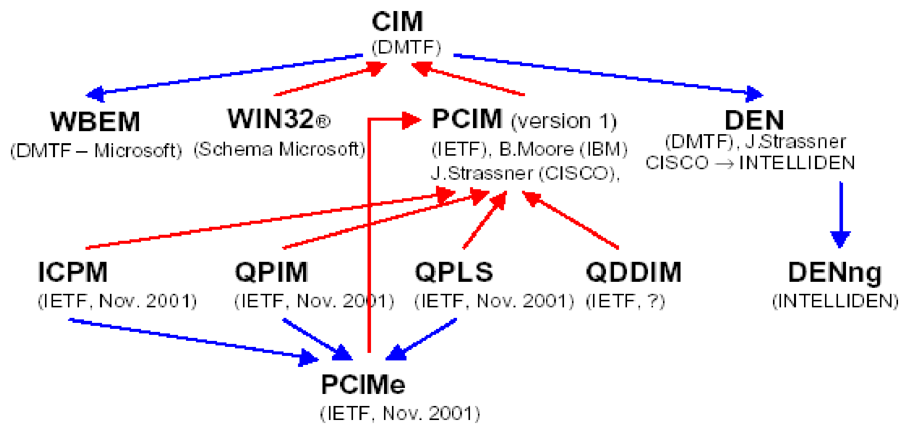

Figure 2. Policy Models. Obtained from the Euronetlab [Corre]

Other highly interesting works for this Thesis are those ones carried out to define a mapping of the policy information model to a shape that can be implemented in a directory that uses the version 3 of the Lightweight Directory Access Protocol (LDAP) as access protocol. This model defines two object class hierarchies: the structural classes that represent information to present and control the data about the policies specified in the PCIM and the relation classes that indicate how the structural classes' instances are related among them.

In the LDAP scheme classes to improve the execution of the interactions that the user establishes in the LDAP server are also added when a great amount of information related to policies must be recuperated. These classes only exist in order to optimise the LDAP recuperations, that is to say, in the policy information model there are not any correspondent classes.

The PCIM classes mapping to an LDAPv3 directory is specified in [Strassner 02] and in [reyes03] a LDAP scheme based on the extensions of the PCIM model is defined.

Policy Core LDAP Schema (PCLS)

PCLS [Strassner02] takes as its starting point the object-oriented information model for representing information and controlling policy data as specified in [Moore01].

PCLS defines the mapping of information model classes to a directory that uses LDAP as its access protocol. Two types of mappings are involved:

1. For the structural classes in the information model, the mapping is basically one-for-one: information model classes map to LDAP classes, information model properties map to LDAP attributes.
2. For the relationship classes in the information model, PCIM's relationship classes and their properties are mapped in three ways: to LDAP auxiliary classes, to attributes representing distinguished name (DN) references, and to superior-subordinate relationships in the Directory Information Tree (DIT).

Implementations that use an LDAP directory as their policy repository and want to implement policy information according to [Moore01] shall use the LDAP schema defined in PCLS, or a schema that subclasses from the schema defined in PCLS. The use of the policy information model as the starting point enables the inheritance and the relationship class hierarchies to be extensible, such that other types of policy repositories, for example relational databases, can also use this information.

An extension of PCLS is the PCLSE [Reyes03], which is a mapping of the PCIM extensions to an LDAP scheme. Appendix I shows more details about PCLS and PCLSE. Also there is a scheme of the different classes and associations that PCLSE use.

2.2 OMG Policy System Architecture

The standard architecture oriented to objects for applications (CORBA) defined by the OMG [OMG] defines the necessary interfaces to support the interoperability and the transparency to build applications distributed between platforms from different manufacturers both the hardware and the software ones, and in the same way, it provides use facilities to customers and servers in several languages.

CORBA separates the object interface from the object implementation obtaining interoperability in the hardware platform and in the software one. Therefore, the management based on CORBA allows the fact of reusing the management service implementations for the future applications appearing in environments based on heterogeneous platforms with distributed processing and oriented to objects.

There are several management environments that own interoperability with CORBA platforms [Mazumdar96]. The distributed object computation that uses a methodology oriented to objects to build distributed applications provides efficient solutions to the heterogeneous network management, for this reason distributed object computation is more and more oriented to the Object Request Broker (ORB) concept. The ORB establishes the communication between the remote objects and the local ones taking into account the aspect that the application is not affected by the low-level infrastructure or the communication details.

All these frameworks provide the remote service invocation in a transparent way and they make easier the fact that the management processing and the services are not found in centralised locations but in distributed ones throughout the network. In this way, the management tasks are delegated to intermediate entities that release some of the management tasks to the central entities.

In general, CORBA is useful as an integration tool for heterogeneous network management domains. The problem lies on the aspect that CORBA distributes static objects and it does not allow the code mobility, requiring a dedicated support as well that in the execution time can be very high. All network devices cannot offer this support required by CORBA. Therefore, the development area in this platform is restricted.

2.2.1 CORBA policies for QoS

The QoS management functions emerged in CORBA [CORBAv3] from the version 3 where the policy-based QoS framework was defined. CORBA Messaging Specification [CORBA-MSG] explains how QoS policies work on the client and server sides.

The QoS policies affect the time-out, the asynchronous invocation routing, priorities, processes at real time, etc. CORBA QoS framework was defined as a set of basic policies to satisfy some services. Either the customer or the server can assign the QoS policies values, which are classified into four levels: system level policies, ORB level ones, Thread level ones and object level ones. Figure 3 shows the four levels at which a CORBA client application can establish policies.

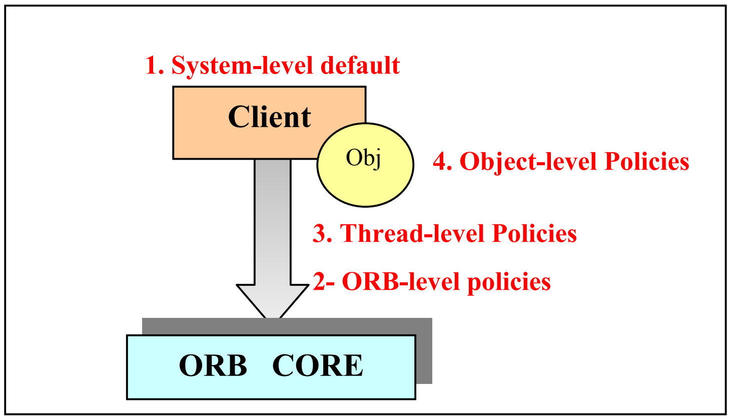

Figure 3. Policy Levels of CORBA

Below, we outline each of these policy levels, ranging from the most coarse-grained control to the most fine-grained control.

System-level default. ORB implementations supply a set of system-wide policy defaults that apply if they are not overridden at any of the levels described below. Because system-level QoS policy defaults are not standardized, portable applications concerned with QoS must be sure to override the desired QoS policies at the appropriate level.

ORB-level policies. By setting policies on an ORB instance, the client can override system-level defaults to control the QoS for all requests made through that ORB. When a client obtains an object reference, that reference is associated with the ORB instance used to make the request. Object references can be obtained either through ORB::resolve initial references, through ORB::string to object, or by receiving the object reference from another operation invocation. If applications use multiple ORBs, they can apply different QoS policies to different invocations. For example, an application might want to communicate with one set of objects using traditional synchronous invocations, but communicate with another set of objects using asynchronous invocations. Such an application might opt to use two different ORBs, with different QoS policies, to communicate with these two sets of objects.

Thread-level policies. Applications can override ORB-level and system-level policies on a per-thread basis. This provides a finer granularity of control that allows requests made in a given thread to have different QoS characteristics than requests made by other threads in the same application process. Applications can override thread-level policies by retrieving a PolicyCurrent object from the ORB's resolve initial references bootstrapping operation and invoking appropriate QoS framework operations, such as set policy overrides.

Object-level policies. The finest level of QoS granularity control available to applications is on a per-object reference basis. The CORBA::Object interface supplies operations to override thread, ORB, and system-level QoS policies, as well as to query the effective client-side policy in effect for a given policy type. The ORB implementation computes the effective policy value by considering the system-level default and applying any overrides from the ORB, thread and object-levels.

The QoS management consists of three types of policies basically:

1. Policies to specify the request lifetime. These policies indicate the initial time and the final time of these requests and of the answers that the server must carry out.
2. Routing policies. All values that these policies can assume indicate the QoS that must be followed through the package route. In case a QoS (higher than the physic configuration that a network must provide) is assigned to a routing policy, then the invocation fails.
3. Queue order policies. These policies are used to assign priorities to the order in which a destination router directs (forwards) the requests to the server. This does not affect the order in which severs process the request. There are four values that can be configured in this policy and the case where the requests are served according to the way in which they are received is considered as the default value. Other possibilities are that the user does not worry about the order, that the requests are ordered according to the priority values that the user establishes or that the requests about to expire pass to the queue front to be attended.

The information about events and actions that the policy must implement and the information of when and how to apply the policy must be distributed in the network elements. In this way, The CORBA objects react when the events are received and they apply the specific management actions. A CORBA advantage is the fact that the subjacent software components can be exchange without having to re-implement the management system or re-specify the existing policies.

2.2.2 Management of a system based on CORBA policies

A network scenario in which the network management is enabled by means of CORBA policies is described in [Reyes01]. This mentioned scenario is part of the GIRIN project supported by the Centro de Investigación Científica y Tecnologíca (CICYT) with reference TIC 200-1042.

The CORBA platform defines the interfaces related to the Quality of Service and interfaces related to routing in order to build the distributed applications that the proposed policy system uses [Reyes02]. This proposed system uses CORBA policies to manage the routing with Quality of Service restrictions and admission control within an ISP. CORBA applications use the name service traditionally to store and recuperate object references. However, in the proposed scenario, a LDAP version3 directory service is used. The CORBA application uses the directory as a repository for the object references, providing a service administrated centrally and replied to be used by distributed applications throughout the network. The network scenario used is shown in figure 4.

The system consists basically of two components: an Admission Control module and a module in charge of providing the route through which a specific kind of traffic must travel in the network, both modules are controlled by CORBA policies (see figure 5). Provided the fact that policies from both modules (Admission Control and Routing) are totally orthogonal, conflicts cannot appear among policies. However, runtime conflicts are solved defining general precedence rules for the policies.

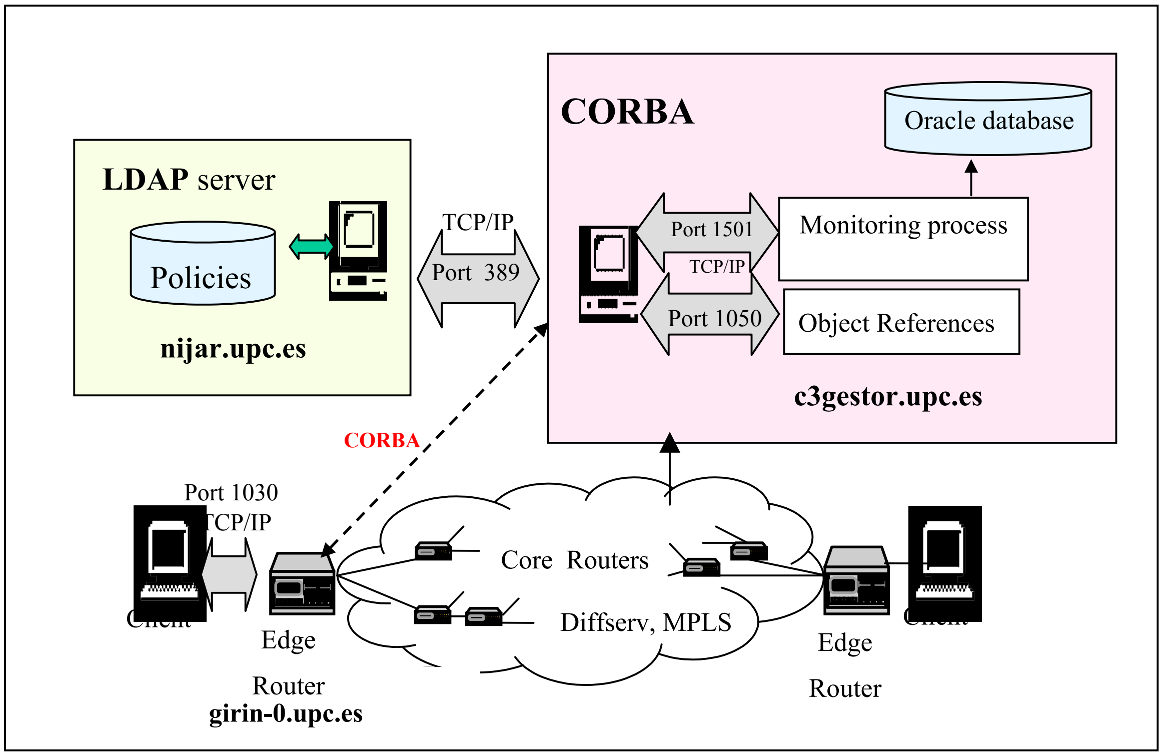

Figure 4. Network Scenario

All the four levels of policies that the CORBA specification establishes are used in the proposed system: system level (default policies), ORB level, thread level and object level [CORBA-MSG]. The proposal is scalable and it allows the fact of increasing both the number of services and the number of policies. The user assigns the convenient QoS value to some policies and, in this way, invocations of a CORBA object particular instance are controlled, which means a high rate of control. However, the server establishes value ranks for the policies that the user must respect.

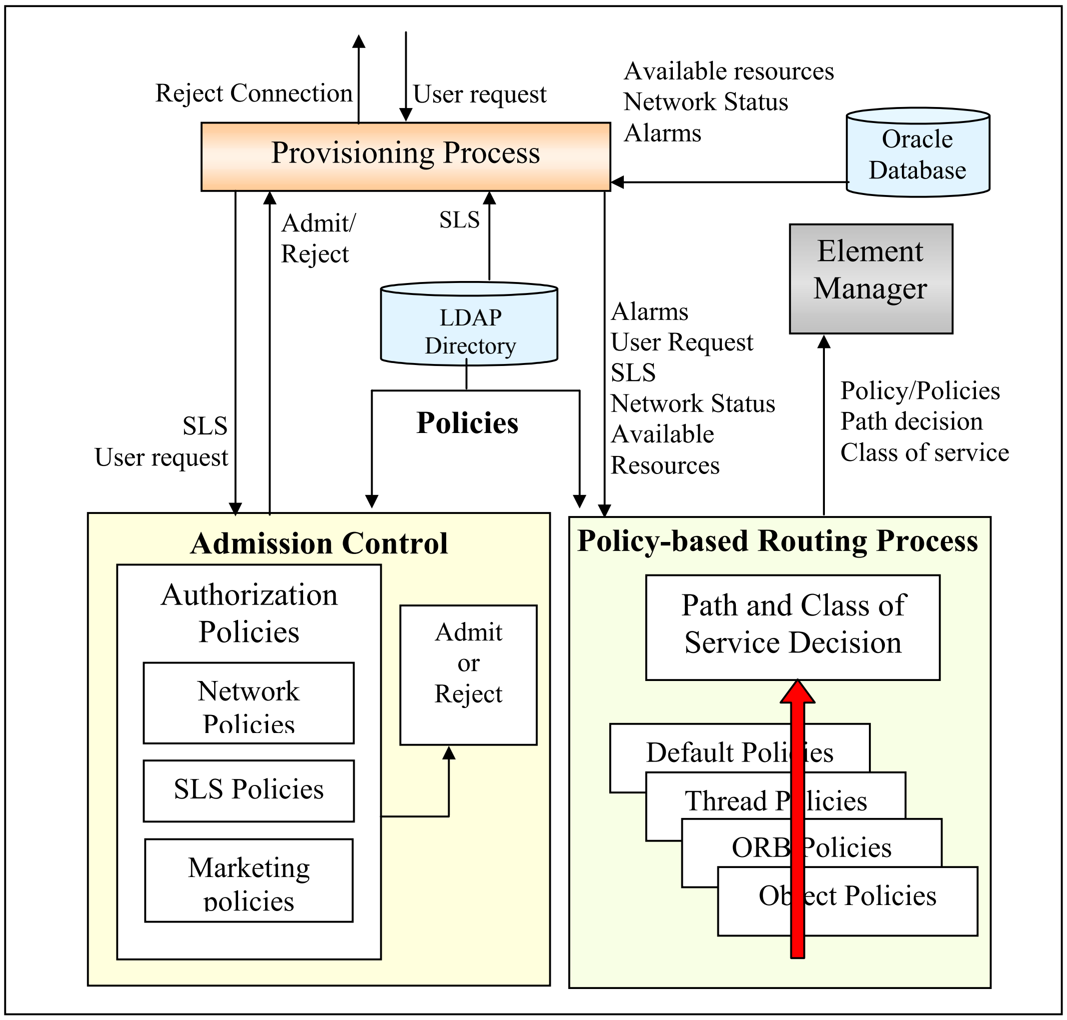

Figure 5. Main modules in the CORBA Application

The proposed management system is built from these four policy levels, taking into account the fact that default policies own the least priority, that is to say, these policies are only applied in case any of the other policies belonging to the rest of levels can be applied [Reyes02]. The previous figure shows both the general scheme of the two basic components of the system and the way in which the policy selection process is carried out among the four levels. For example, the object-level policies overwrite thread-level ones and successively. This model avoids conflicts among the levels of the policies and reduces the number of conflicts among policies of the same level.

2.2.3 CORBA policies for Intelligent Buildings

This section presents the previous management platform applied to the Intelligent-Buildings environments. First of all, Intelligent-Building has been defined as "one that utilizes computer technology to autonomously govern the building services environment so as to optimize user comfort, energy-consumption, safety and work efficiency" [Callaghan00].

From the telecommunication point of view it is very important for IBs to consider communication between intelligent autonomous systems, intelligent personal devices, management information systems, mobiles, etc. In addition it needs to support communication between different Intelligent Buildings.[Reyes-IB].

This research presents a network architecture model for IBs, which seeks to achieve the best form of communication and management both internally within an IB and externally between IBs. Full integration of IBs with telecommunication systems is becoming increasingly important, for example to facilitate more efficient commercial transactions.

Considering the nature of intelligent environments [Colley01], some open network architectures were established to allow a free conversion between protocols and high speeds in communication. This is the reason why the research justifies the use of open technologies such as the Universal Mobile Telecommunications System (UMTS), the Internet protocol version 6 (Ipv6) and platforms for the management of the different environments, such as CORBA and Jini.

The policy-based CORBA management platform is efficiently applied to the domotic network environments due to the fact that those systems consist of several and multiple devices, compute equipments and software systems that must interact among them. All devices in an intelligent building (central-heating, security systems, lightning system, washing machines, refrigerators, alarm systems, etc.) own agents with a low process capacity and a much reduced store space [Cayci00]. This is the reason why the policy-based management is extremely efficient.

The management architecture of the different networks involved in intelligent buildings, both for the control inside the building and for the interconnection with other fix and mobile networks, is exposed in [Reyes-IB]. In those systems it is very important to specify clearly the policies to apply them correctly. The devices technology evolves quickly and the fact of scalability and an easy integration of new nodes in the network are very important characteristics to be considered in the domotic systems. In case devices in a building are aggregated, eliminated or changed, then the policy repository must be updated to adapt the network to the new changes.

Policies are stored in an independent repository from where only the necessary policies applied directly by the agents distributed in the devices are recuperated. Another important fact is that the CORBA priority model is based on policies and it allows the assignment of different precedence levels to the services and tasks that an intelligent building offers. CORBA was also chosen as the management architecture for the intelligent buildings due that it allows that every intelligent environment owns different devices, calculation equip and software systems because all this heterogeneity does not represent a problem.

CORBA resolves the interconnection of all rooms (physical and logical units in a building) together with their corresponding connected devices, defining interfaces and services to support the interoperability between the heterogeneous nodes that form the building and offering a high level of transparency to build the distributed applications that solved all the needs in intelligent buildings. The interconnection and management in intelligent buildings with other intelligent interactive environments is more and more complex due to the great variety of devices they use and the different functions and services that offer. This problem becomes easier thanks to the use of CORBA, because it allows the interoperability of the different devices, software, protocols, etc. that intelligent buildings own.

The following figure shows the different nodes and technologies that interact in the management platform proposed for domotical environments, mainly for those intelligent buildings that exist in the group of investigation about intelligent buildings at the Department of Computation in the Essex University [Essex].

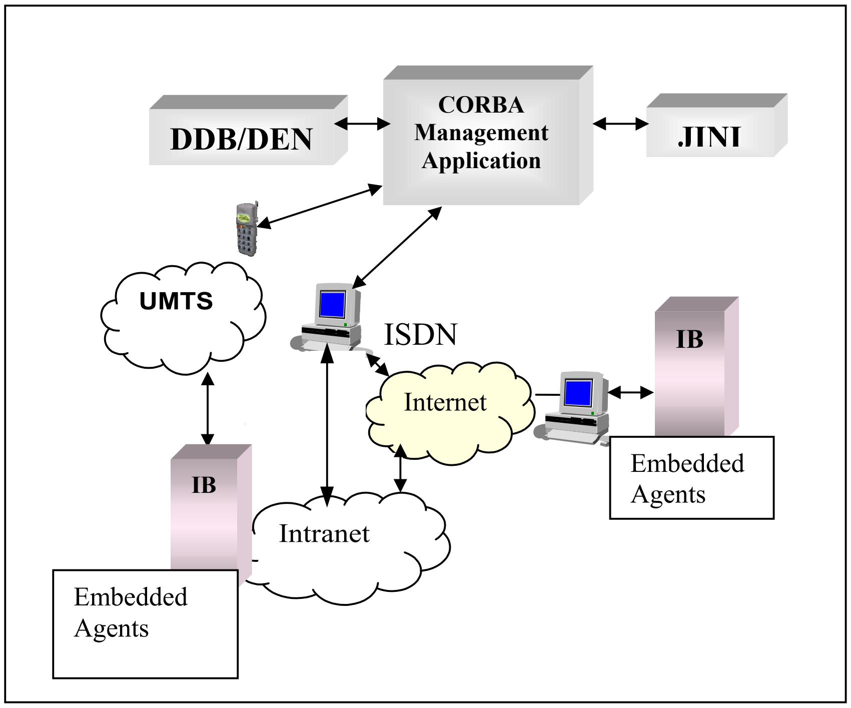

Figure 6. IB Interconnection Management Platform

2.2.4 CORBA policies for OSA/PARLAY

The Parlay Group [Parlay] is an open multi-vendor consortium formed to develop open technology-independent application programming interfaces (APIs) enabling: technology, Internet and e-Business companies, independent software vendors (ISVs), software developers, network device vendors and providers, service bureaus, application service providers (ASPs), application suppliers, and large and small enterprises to develop applications and technology solutions that operate across multiple networking platform environments.

Nortel uses UMTS network applications. Vodafone, among other enterprises, is using the CORBA architecture to accede to the network services. This architecture can be integrated in intelligent buildings systems and in the policy-based management systems defined in this Thesis.

2.3 Contributions in this chapter

Next we list the main contributions of this chapter.

1. Management of a routing resolution and admission control system based on CORBA policies. This system regards a conflict resolution module between the CORBA policies.
2. Use of the CORBA platform and UMTS for interconnection, communication and management of intelligent buildings with mobile autonomous environments.
3. Distribution mechanisms carried out by CORBA protocols, see section 3.3
4. The distributed management oriented to CORBA domains according to compatible and normalised ORBs and IDLs is adequate before a high level of ad-hoc network nodes in an intelligent building and to monitor a whole of sensors or transducers in a fix network. The network architecture proposed provides great facilities in the interconnection among fix networks and wireless networks.
5. The use of mobile agents based on Java for management functions in remote equipments (especially in case of wireless nodes) was analysed.
6. Grouping of policies into roles depending on the characteristics of the system to be managed.
7. PBMS integration as management according to TMN network

References

Papers

[Barba02] Antonio Barba, Ernesto Sánchez. *An Architecture for Active Network Performance Management Based on Intelligent Agents.* Mobile Agents for Telecommunication Applications (MATA), Proceedings. Lecture Notes in Computer Science 2521 Springer 2002, ISBN 3-540-00021-6 Barcelona, Spain. 2002 pp. 94-104

[Brunner01] M. Brunner, J. Quittek, "MPLS Management using Policies" , Proceedings of the 7th IEEE/IFIP Symposium on Integrated Network Management (IM´01), May 14-18, Seattle, USA, 2001.

[Callaghan00] Callaghan, V., Clarke, G., Pounds-Cornish, A., Sharples, S. Buildings as Intelligent Autonomous Systems: A Model for Integrating Personal and Building Agents. 6th International Conference on Intelligent Autonomous Systems, Venice 2000

[Cayci00] Cayci, F., Callaghan,V., Clarke, G., A Distributed Intelligent Building Agent Language (DIBAL) ISAS 2000

[Clarke00] Clarke, G., Pounds-Cornish, A., Callaghan, V., Intelligent Habitats: The Interaction of People, Agents and Environmental Artefacts 4S/EASST 2000, Vienna, Austria, September 2000

[Colley01] Colley, M., Clarke, G., Hagras, H., Callaghan, V., Pounds-Cornish, A. Intelligent Inhabited Environments: Cooperative Robotics & Buildings In 32nd International Symposium on Robotics (ISR 2001), Seoul, Korea, 2001.

[Hamada98] T. Hamada *Role-based Access Control in Telecommunication Service Management - Dynamic Role Creation and Management in TINA Service Environment*. ACM Workshop on Role-Based Access Control 1998 pp. 105-113

[Lupu97] Lupu E. Sloman M. *A Policy Based Role Object Model*. First International Enterprise Distributed Object Computing Workshop (EDOC'97), IEEE. Australia, Oct. 1997. pp. 36-47.

[Lupu99] Emil Lupu, Zoran Milosevic, Morris Sloman *Use of Roles and Policies for Specifying and Managing a Virtual Enterprise.* Research Issues in Data Engineering (RIDE) IEEE-CS 1999. pp. 72-79

[Mazumdar96] S. Mazumdar. *Inter-Domain Management between CORBA and SNMP*. International Workshop on Distributed Systems: Operations & Management, IFIP/IEEE L'Aquila, Italy, October 1996.

[Reyes01] Angélica Reyes, Antoni Barba. *Arquitectura de un sistema de gestión basado en políticas de servicio en Internet 2.* Primer Congreso Iberoamericano de Telemática CITA2001. Cartagena de Indias, Colombia. 2001

[Reyes02] Angélica Reyes, Antoni Barba*, Architecture of a Policy System for Routing Management and Admission Control.* Eunice / IFIP 2002. Noruega

[Reyes 03] Reyes, A., Barba A., Moron, D., Brunner, M., Pana M. *Policy Core Extension LDAP Schema (PCELS)*, Internet Draft. February 2003.

[Reyes-IB] Reyes, A. Barba, A., Callaghan,V., Clarke, G. The integration of Wireless, Wired Access and Embedded Agents in Intelligent Buildings. SCI2001. Florida, USA.

[Salceda02] J. Salceda, D. Fernández, R. Doallo, et.al. Gestión de Sistemas Distribuidos mediante Repositorios LDAP Symposium On Informatics And Telecommunications September (SIT) 2002, Sevilla, Spain

[Schmidt00] D. Schmidt, S. Vinoski. *Object Interconnections. An Overview of the OMG CORBA Messaging QoS Framework.* C++ Report magazine..March 2000

[Strassner02] J. Strassner, B. Moore, R. Moats E. Ellesson. *Policy Core LDAP Schema* Internet-Draft. October 2002.

Standards

[Blake98] S.Blake, D. Black, M.Carlsson, E.Davies, Z.Wang, W. Weiss. *An Architecture forDifferentiated Services*. IETF Request for Comments (RFC) 2475. December 1998

[Braden94] R. Braden, D. Clark and S. Shenker. *Integarted Services in the Internet Architecture: an Overview.* IETF Request for Comments (RFC) 1633, June 1994

[Kent98] S. Kent, R. Atkinson. *Security Architecture for the Internet Protocol.* IETF Request for Comments (RFC) 2401. November 1998.

[Yavatkar00] R. Yavatkar, D. Pendarakis, R. Guerin. *A Framework for Policy-based Admission Control*. IETF Request for Comment (RFC) 2753. January 2000.

[Moore01] Moore, E. Ellesson, J. Strassner, A. Westerinen. *Policy Core Information Model- Version 1 Specification.* IETF Request for Comment (RFC) 3060. February 2001.

[Westerinen01] A. Westerinen, J. Schnizlein, J. Strassner, et. al. *Terminology for Policy-Based Management.* IETF Request for Comment (RFC) 3198. November 2001.

[Moore03] B. Moore, Ed.. *Policy Core Information Model (PCIM) Extensions.* IETF Request for Comment (RFC) 3460. January 2003.

[Hodges02] Hodges, J., and Morgan R., *Lightweight Directory Access Protocol (v3): Technical Specification*, IETF Request for Comment (RFC) 3377, September 2002.

[Durham00] D.Durham, J.Boyle, R.Cohen, S.Herzog, R.Rajan, A.Sastry. *The COPS (Common Open Policy Service) Protocol.* IETF Request for Comment (RFC) 2748. January 2000

[Chan01] K. Chan, J. Seligson, et.al. *COPS Usage for Policy Provisioning (COPS-PR).* IETF Request for Comment (RFC) 3084. March 2001.

[OMG] Object Management Group, CORBA Messaging Specification, OMG Document orbos/98-05-05 ed., May 1988.

Web Pages

[CIM] Common Information Model (CIM)
http://www.dmtf.org/standards/standard_cim.php

[CORBAv3] CORBA versión 3 http://www.omg.org/news/pr98/compnent.html

[CORBA-MSG] Object Management Group, *CORBA Messaging Specification*, OMG. Document orbos/98-05-05 ed., May 1998.

[Corre]O.Corre, V. Ksinant *Policy Modeling in a PBM.* Architecture *6WIND / Euronetlab http://www.euronetlab.com/seminar/corre_policy_overview.pdf*

[DEN] http://www.dmtf.org/standards/standard_den.php

[Essex] http://cswww.essex.ac.uk/

[IETF-1] IETF Policy Framework Working Group, http://www.ietf.org/html.charters/policy-charter.html

[IETF-2] IETF Resource Allocation Protocol (RAP) Working Group, http://www.ietf.org/html.charters/rap-charter.html

[Parlay] Grupo OSA Parlay: www.parlay.org