Appendix II

Policy system implementation

This appendix is a summary about the implementation of processes involved in the proposed policy-based management system. The Linux system was used taking into account its advantages to provide traffic control offers. It supports several classification methods, priorities, partition and limitation in relation to the ingress and egress traffic, in a management module of Differentiated Services.

First, the necessary process for the network node configuration will be explained in detail; second, the monitoring process that stores its results in a relational database will be exposed. At the end of this appendix there is the agent implementation of the routers and the possible routing ways that the proposed policy system has.

II.1 Network node configuration

A scalable IP address assignation is created to configure the network nodes and in this way, a package is able to reach its destination through different paths, depending on its class of service (CoS) and the network status.

The configuration varies depending on the fact that they are interfaces on the customer's side or interfaces on the network side, which also varies depending on the fact if they are edge or core routers.

Core routers send the packages through the interface indicated by the routing chart taking into account de destination address. The edge routers need a special configuration in order to assign a route to a customer. In the implementation, the third address octet was chosen to establish the route. Example: 192.168.3.1 corresponds to an address in the route 3.

The fourth octet is used to differentiate the customers' addresses that use the same route and in this way, every customer will have a different core network address. Therefore, the number of addresses corresponding to the number of paths and to the number of customers that use those paths is configured by an interface.

The network consists of three PCs and two Sun equipments (Girin7 and Girin0) that accomplish the functions shown in the following table.

Girin-7: Edge Router1= RF1
Girin-3: Edge Router2= RF2
Girin-6: Core Router3= RC3
Girin-0: Core Router1= RC1/Customer

Table 1. SUN equipments functions.

The configuration process is divided in three stages that can be graphically seen in the figure 2 of chapter 3. Every component in the configuration process is detailed as follows.

Traffic control. It classifies the ingress traffic depending on the class of service to which it belongs, which results in the fact of having aggregated flows (BA) instead of having individual flows (MF-microflows). The traffic control configures the police and shaping functions for every flow (MF) and determines what to do with the exceeding traffic. In that case, it can drop the packages or remark them to another class of service.

In the implementation presented in this Thesis, the exceeding packages are remarked to a Best Effort class of service. In future implementations, the exceeding traffic with AF service class can be remarked considering the same class (AF1, AF2, AF3 or AF4) but changing the package drop precedence level. The standard way considers 3 precedence levels for each of the four classes that the AF service owns. [Grossman02]

Routing. This module forwards the packages existing in an individual flow through a route that the path selector process determines (explained in chapter 5) and through the admission control.

The routing block develops three basic functions:

1. Translation of the IP address of the destination network into the destination address of the core route (DNAT module).
2. Router exit interface election. The exit interface is chosen considering the destination address of the core route. (Kernel of Linux carries out this function being based on the static routing table previously configured).
3. Translation of the IP address of the origin network into the origin address of the core route (SNAT module).

Exit. This module configures the router core network exit interfaces carrying out a common flow traffic control for every class of service and marking the DS field depending on the corresponding class. Then the traffic flow is allocated on the queue.

The exit block in an edge router develops the following functions:

1. Police function for BA. The individual flows in the same class form an aggregated flow (BA) that is controlled by means of police functions based on the algorithm Tocken Bucket. The parameters for this algorithm are the bandwidth and the bucket size (Burst), that are equivalent to the available bandwidth for every service or BA and the exceeding traffic.
2. Marking. An IP package Ds field marking is provided by a DSMARK discipline that includes the Traffic Control of Linux. This field value depends on the package class of service.
3. Package sticking. It is based on the HTB discipline (Hierarchical Token Bucket) to create different sticking disciplines depending on the classes of service. In this way, the PHB EF is associated with a queue discipline called FIFO; in case of Best Effort, a RED discipline (Random Early Detection)is associated, which drops the packages at random when they surpass a pre-established threshold; in case of AF PHB, a GRED discipline (Generic Random Early Detection) is associated, which creates different virtual queues in the same class depending on the packages elimination priority.

Different priorities are considered, so that a low-priority class package will not be able to be served until all superior priority class packages have been served.

Core routers configuration only includes aggregated flows (BA) and routes the packages depending on the class to which it belongs. Core routers own a routing module to obtain the destination IP address of the entrant package and send it to the exit interface that indicates its previously configured routing table. In the same way, it owns an exit module that is installed in every core network exit interface. It is very similar to the edge router exit

module, except the fact that the DSMARK discipline does not mark the DS field of the packages, but it detects it already marked and it sends it to the corresponding queue in accordance with its service class.

II.2 Monitoring process Implementation

This process analyses the routers status and the parameters that are obtained are dynamically stored in a database.

In this implementation, monitoring is carried out by means of the Girin-0 equip. However, the system is prepared to the fact that monitoring can be developed in a distributed way too.

The main applied technologies for this module are the JDBC data Access, API of Sun, for the communication with the database Oracle, the JDMK (Java Development Management Kit) of Sun, for the communication with the routers agents, and the Threads of Java to create concurrent processes.

HtB scheduler statistics are obtained by means of a Perl script and calling the Traffic Control system. This call to the system provides statistics for every queue discipline associated to every service, and statistics of the discipline associated to the whole link. The most meaningful values that are extracted for the system monitoring are the number of bytes that every queue sends, the number of packages that are sent, the packages that are lost and the packages that are on the queue.

The script generates a core timer and it emits shows about these data establishing a certain period of time between modifiable statistics, which is a default second. The following parameters are calculated:

Available bandwidth in the link. It is calculated as the existing difference between the occupied bandwidth and the total available bandwidth. It is obtained taking into account the

monitoring timer (T) and the bytes that every queue sends (B). R [KB] = Bi – Bi-1 / Ti – Ti-i where i= show

Delay in the queue packages. It is calculated as the addition of all delays existing in every node. In the nodes, the delay is measured as the time in which the packages are on the exit queue. It is assumed that there are not nor any propagation delays or processing ones in the routers neither delay caused by collisions in the hub. In order to calculate it, the number of sent packages (N), the number of sent bytes (B), the number of packages on the queue (C) and the bandwidth (R) are used.
Average package length (L) [bytes] = N [packages] / B [bytes] <1500 bytes = maximum ethernet package length.
Delay (D) [ms] = (C [queue packages] * L [bytes] / R [KB]) * 1000 in milliseconds.

Jitter. It is calculated as the existing delays difference between consecutive packages on the same queue. The maximum jitter introduced by the different routers that are involved in the route is considered. There are different ways of measuring the jitter in a route. In the implemented scenario, when connections at real time are considered, the metric proposed in [Demichelis02] and a first-order profitable exponential estimator 1/16: Jitter[i] = 15/16 * Jitter[i-1] + 1/16 * Jitter[i] where i=show are used.
To connections that do not happen at real time, the delay is obtained form the current show (Di) and the delay from the former show (Di-1), considering the first-order estimator, which reduces noise:

1. JR[i]= | Di – Di-1|
2. JR[i-1]= 15/16 * JR[i-1] + 1/16 * JR[i] in milliseconds, where i= show

Losses of queue packages. Only those losses of packages on the router exit queues are considered. In future works the effect of the losses happened by communication mistakes could be considered. This calculation is extracted from the number of the lost packages (P) and the number of sent packages (N). Losses are expressed as a percentage of the received packages in respect of the sent ones: Loss = P/N * 100

The following figure shows the different levels in which monitoring identification carried out:

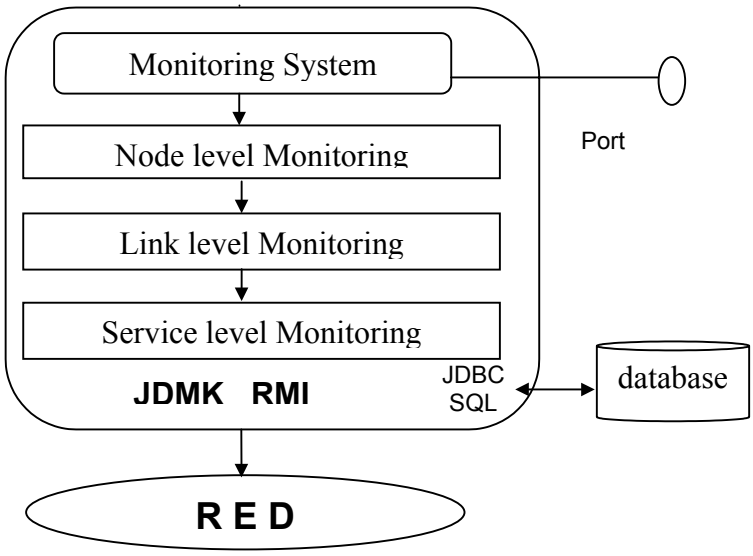


Figure 1. Monitoring system for different levels

II.3 Oracle database Parameters

The database that was installed in the Girin-0 equip for the policy system is the 8.1.5 version of the relational database Oracle. Its function consists of storing the network status information, as for example, the existing nodes and their interfaces, the classes of service, the pahts between network ends, customers, network connections, etc.

The entity-relation diagram followed to define the database structure is presented as follows. The grey squares represent the intermediate relational tables that the entities with a n-to-m relation need. PLSQL developer and SQLPLUS were used to create and list tables.

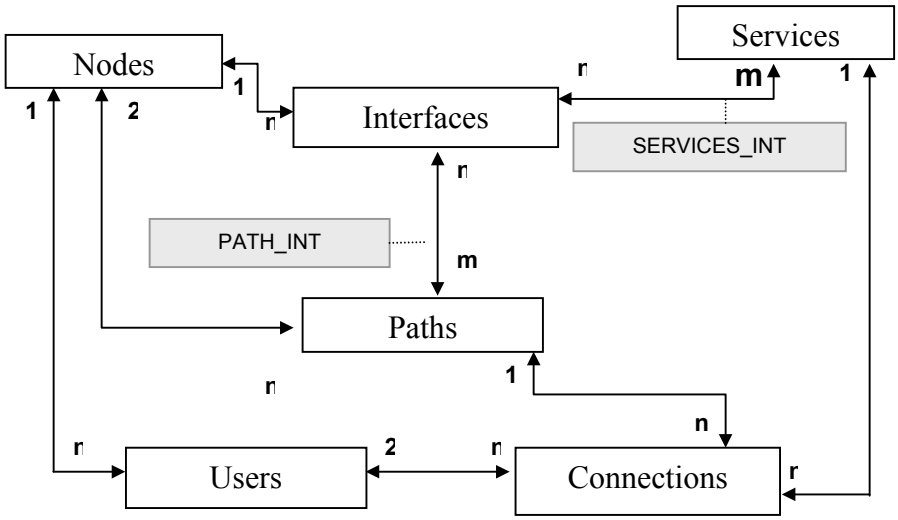


Figure 2. Entity-relation diagram of the policy system database.

Entities and tables

The implementation carried out until now owns six entities and the tables of the database are next exposed. Considering the fact that the system is scalable, more entities and tables can be added to the system

Nodes. This table stores general information about the network nodes: identifier, name, description and router type (edge or core).

	ID	NOMBRE	DESCRIPCION	TIPO
▶ 1	0	GIRIN-0	SPARC ULTRA 5. S.O: SOLARIS	RI
2	2	GIRIN-2	SPARC ULTRA 5. S.O: SOLARIS	RI
3	3	GIRIN-3	PC PENTIUM II S.O: LINUX 2.4.18	RF
4	6	GIRIN-6	PC PENTIUM II S.O: LINUX 2.4.18	RI
5	7	GIRIN-7	PC PENTIUM II S.O: LINUX 2.4.18	RF

Table 2. Example of the node table.

Interfaces. They represent the network links and store the following information: an identifier, the name of the interfaces in each machine, the node identifier to which the interface belongs, the link total bandwidth, the bandwidth occupied in that very instant and the type of interface (core or edge).

	ID	NOMBRE	NODO_ID	BW_LINK	BW_OCUPADO	TIPO
▶	1	eth1	7	250	11,72	out
	2	eth2	7	500	0	out
	3	eth1	3	1000	0,508	out
	4	eth2	3	100	0	out
	5	eth0	6	10000	0,631	out
	6	eth1	6	10000	13,58	out
	7	qfe1	0			out
	8	qfe2	0			out
	9	qfe3	0			out
	10	qfe1	2			out
	11	qfe2	2			out
	12	qfe3	2			out

Table 3. Example of the interface table.

Services. The different types of services that are implemented in the application and in the network are stored in the database. Its fields are: an identifier, the name of the class of service and its description. Every interface can own one or more classes of services.

	ID	NOMBRE	DESCRIPCION
▶	1	Expedited Forwarding	
	2	Assured Forwarding Clase 1	
	3	Assured Forwarding Clase 2	
	4	Best Effort	

Table 4. Example of interface table.

There is an auxiliary table that establishes the relation between the interfaces and the services with the following fields: a relation identifier, an interface identifier and another for the service, the service bandwidth in this interface, the total reserved bandwidth, the bandwidth occupied in that instant, the queue delay of this service in that interface, the jitter and the losses level.

	ID	INT_ID	SERVICIO_ID	BW_SERVICIO	BW_RESERVADO	DELAY	LOSS	JITTER	BW_OCUPADO
▶	1	1	1	150		0	0	38,18403	12,7
	2	1	2	62		0	0	0	0
	3	1	3	62		0	0	0	0
	4	1	4	15		0	0	9,339569	0
	5	2	1	300		0	0	0	0
	6	2	2	125		0	0	0	0
	7	2	3	125		0	0	0	0
	8	2	4	30		0	0	0	0
	9	3	1	600		0	0	0	0
	10	3	2	250		0	0	0	0

Table 5. Example of the auxiliary interface and service table.

Paths. It stores all the possible paths between two network ends. Two basic parameters are considered: the edge node identifier and the intermediate interface identifier through which the path goes.

	ID	NOMBRE	EXTREMO1	EXTREMO2	PRIO
▶ 1	2	Ruta 2	7	3	200
2	3	Ruta 3	7	3	300
3	4	Ruta 4	7	3	400
4	5	Ruta 5	7	3	500
5	6	Ruta 6	7	3	600
6	7	Ruta 7	7	3	100

Table 6. Example of the route table.

This table identifies the route ends, but in order to be able to differentiate the paths that own the same ends, the network core node interface is identified. This is the reason why between routes and interfaces there is a n-to-m relation and an auxiliary table having both the route identifier and the interface one is required.

	ID	RUTA_ID	INT_ID
▶ 1	1	7	1
2	2	7	6
3	3	7	3
4	4	7	5
5	5	2	2
6	6	2	4
7	7	3	2
8	8	3	7

Table 7. Example of the auxiliary route and interface table.

Customers. Network customers are linked to the network ingress point. This table owns an identifier field, the IP address, the host customer's name and a field with the customer's characteristics. The node to which the customer is connected has to be an edge router.

	ID	IP	HOST	NODO_ID	PROPIEDADES
▶ 1	1	147.83.39.72	GIRIN-0.UPC.ES	7	cliente de prueba 1
2	2	147.83.39.74	GIRIN-2.UPC.ES	3	cliente de prueba 2
3	3	192.168.10.2	girin-0.upc.es	7	cliente de prueba 3
4	4	192.168.20.2	girin-2.upc.es	3	cliente de prueba 4

Table 8. Example of the auxiliary customer table.

Connections. The connections established in the network are kept in the database. It can keep only the active connections or, on the contrary, all kinds of connections in order to have a connection file. The different fields in this table are the following: an identifier, an identifier of all customers involved in the connection, on for the communication route and for the communication class of service, bandwidth reserved in the communication, maximum end-to-end delay expected in the communication, maximum jitter, loss maximum percentage and time in which the communication is established.

	ID	CLIENTE1_ID	CLIENTE2_ID	RUTA_ID	SERVICIO_ID	BW_RESERVADO	DELAY	JITTER	LOSS	TIME_START
▶ 1	1	1	2	7	1	100	5	150	0.2	02/01/2003 12:00:00
2	2	3	4	2	3	200	20	75	0.01	02/01/2003 12:05:24

Table 9. Example of the connection table.

II.4 Router agents

Some agents acting as bridges between the remote management applications are used in order to carry out the monitoring and configuration of the routers. The JDMK (Java Dynamic Management Kit) technology of Sun allows the fact that, on creating a generic agent, it can be applied in any network node. JDMK has operations that make easier the control and management application creation in the agent itself, reducing in this way the processing in the remote managers. This function distribution is one of the basic approaches of the new network management platforms and it considerably improves the decrease in the network loading process.

The design of a JDMK agent experiences a first stage of agent resources instrumentation and a second stage of design and programming.

Agent resources instrumentation. Agent resources are those entities, physical or virtual, that are exposed by the agent in order to make them be managed by remote applications. A resource can represent the whole network node or only some parts of it. JDMK

denominates these resources as Managed Beans (Mbeans), because they are based on Java Beans.

A resource management interface (Mbean Interface) must be created in order to instrument a resource. This interface is formed by some attributes and accessible operations for the remote managers, which will be able to accede to the attribute content, modify it and execute the operations.

In an edge agent three are three kinds of resources. They are described as follows:

a. Customers. This Mbean encapsulates the information of a customer that is connected in the router at that very moment. When the network accepts a new connection request, the agent receives a policy action that indicates the configuration that this new edge router must own to satisfy the user's request with the customer's parameters. The agent creates a Mbean with the following information: IP destination address, IP origin address, destination port, origin port, transport protocol, connection route, class of service, burst level, policy action that must be applied in case of an excess of traffic and a maximum rate for a CoS.
b. Interface scheduler. This Mbean encapsulates the scheduler information belonging to the router interface. When the router agent is started automatically, a Mbean is created for each interface with some default values that configure the exit scheduler for the interface. This configuration is carried out by means of shell-scripts and system calls to the Traffic Control of Linux. Every CoS scheduler counts on the following parameters: maximum rate and maximum burst for this class of service and queue length FIFO.
c. Interface statistics. It obtains the statistics from an interface. When the router agent is started automatically, a statistics Mbean is created for each interface. On creating this Mbean, the monitoring process, whose results are the entry parameters for this Mbean, is initiated.

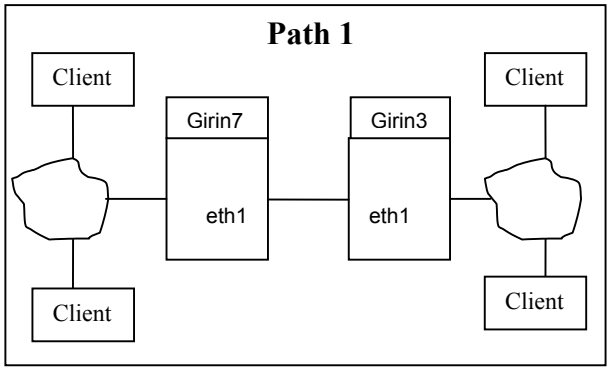
Agent design and programming. The edge router agent initiates an InterfaceStandard and an InterfaceStarsStandard for every exit interface towards the core network. Both edge routers configure the necessary routers for the connection between two customers at the same time. When the agent is started, a port is opened in order to make remote applications be connected. JDMK includes some classes to configure a communication protocol adapter in order to make the agent designer program independently from the communications channel. In the system agents, a RMI communication adapter is initiated and, in this way, JAVA is considered over RMI.

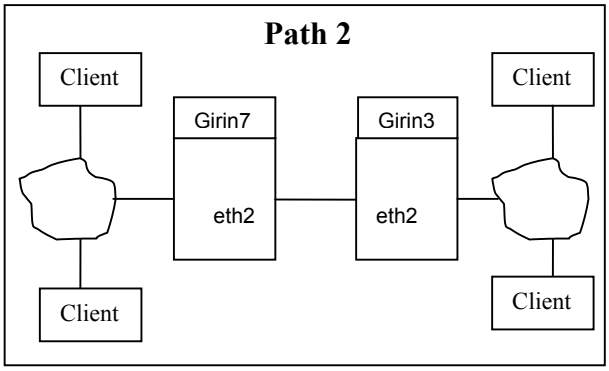
The agent starts a HTML service, which is an interface to visualise all agent Mbeans and its parameters via web. It is important to mention that the core router agents will not own customer Mbeans, because they work with aggregated traffic (BA) and not with customer individual traffic (MF). There are only agents for core routers in Girin-6, because it is the only one that owns a HTB scheduler.

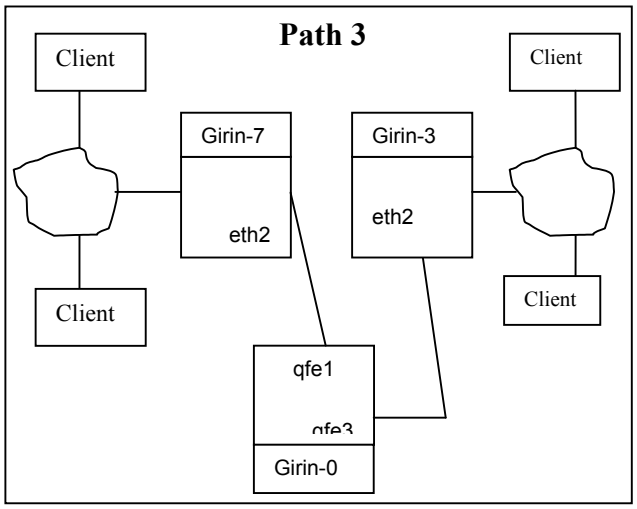
The following quartet is used to differentiate an individual flow (MF) from another: IP origin – IP destination – Origin Port – Destination Port. [MF-CLASSIFIER]

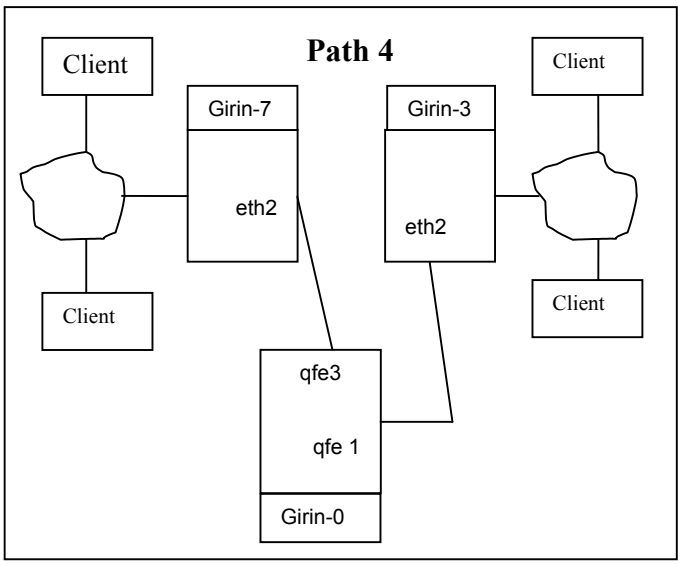
The following elements are configured for every flow: bandwidth in Kbit/s, the exceeding number of bytes that is allowed, the corresponding action for the exceeding traffic (DROP or REMARK) and the class of service (EF-AF1-AF”-BE). The policy system sends these parameters to the node manager in charge of the network devices configuration.

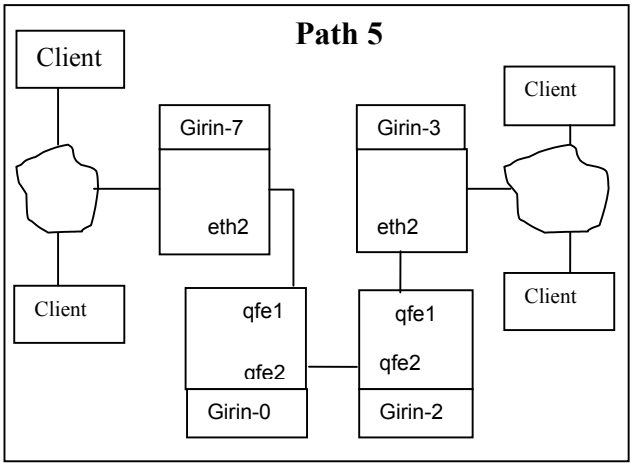
Path schemes

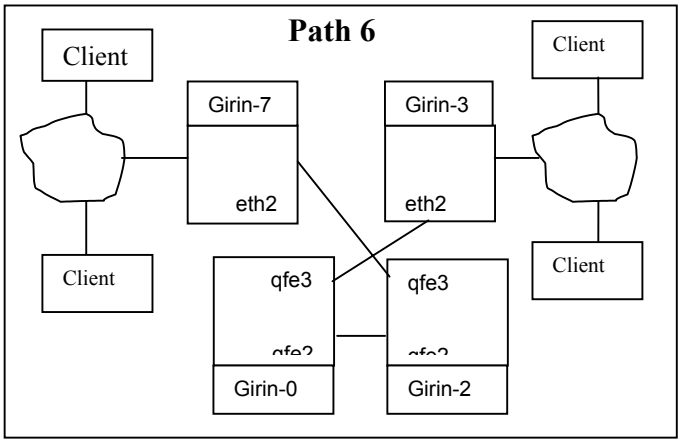


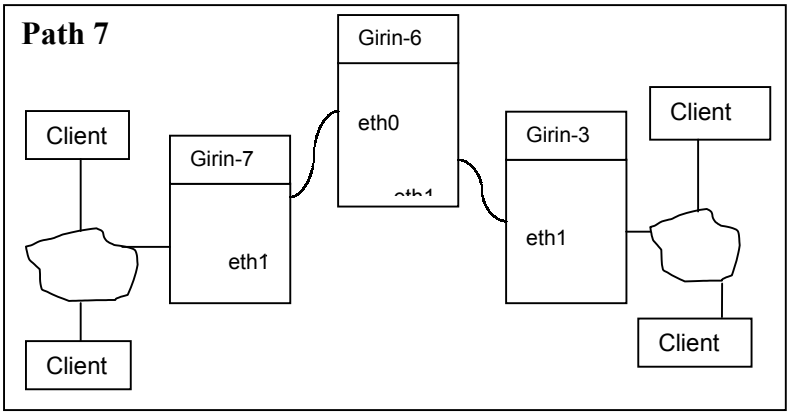







Path 5
Client
Client
Client
Client
Girin-7
eth2
Girin-3
eth2
qfe1
qfe2
qfe1
qfe2
Girin-0
Girin-2


Path 6
Client
Client
Client
Client
Girin-7
eth2
Girin-3
eth2
qfe3
qfe3
Girin-0
Girin-2


Path 7
Girin-6
eth0
Client
Client
Girin-7
eth1
Girin-3
eth1
Client
Client

II.5 Graphical Representation Application

These tools have been designed in order to study the possible conflict resolution in case of overlapping among network and service policies. In this case the geometrical vision helps to distinguish a right design for parameters and policies where are applied.

Every new policy can be associated with some coordinates (or percentages) that each axis owns, defining, in this way, a new model point.

Resulting topology of this disposition is a four-dimension sphere or hyper-sphere. The model, which the program works with, consists of 80 points. The aims this program pretend to achieve are the following:

1. The program must visualize the hyper-sphere, projecting it from the four-dimension space to the three-dimension physic world in order to be able to visualize it in the screen (2D projection).
2. It is necessary to have some navigation tools to move the camera or point of view, and choose the area we want to see and the place from which we look at it. There are two cameras: the 4D one, which projects the sphere to the three-dimensional space, and the 3D one, which visualizes the scene in the flat screen (2D). First the 4D camera is moved until it finds a comprehensive projection around the area to study. Later, the 3D camera is moved to finish the process and the figure is rotated to understand better the image three-dimensional aspect.
3. Placing a point on the centre of the scene to see which its neighbours are in a better way.
4. Adding new points to the model: points that have been computed from the existing ones, for example, average point between two points, barycentre of the triangle formed by three points or the centre of a tetrahedron.
5. Additional functionalities as well, for example, the hypercube visualization (a four-dimension cube, formed by 8 cubic faces), a show of the reference axis, an animated rotation in four dimensions, an animation with the camera zoom or a show only of half the hyper-sphere to simulate the elimination of hidden parts in the scene.

User´s Interface

The execution of the program opens two windows, a text console where the program shows the messages and information to the user and a graphic window where the hyper-sphere (and / or all the operations that the user has executed) is visualised.

The index has next elements: Direct access to the hyper-sphere, V4 Hyper-sphere, Initiating the program, Finished initialisation, the Hyper-sphere.

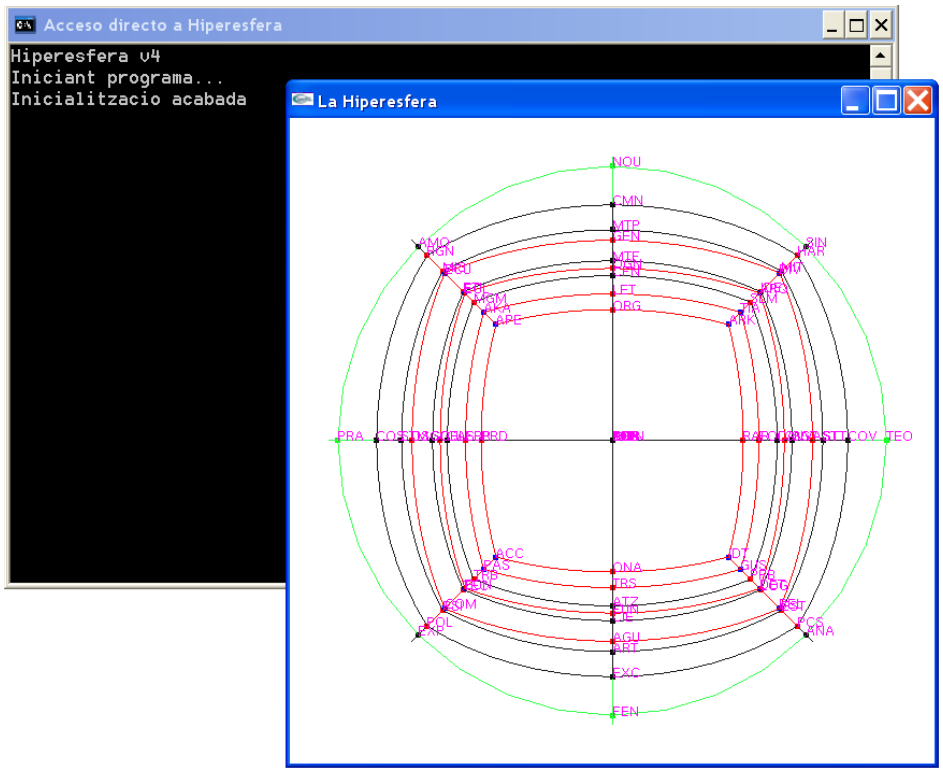


Figure 3. Hyper-sphere

Three-dimensional and four-dimensional navigation are carried out by pressing some keys, while the selection of points is executed by means of the left button of the mouse; with the right button a pull-down menu (with its functions implemented) appear.

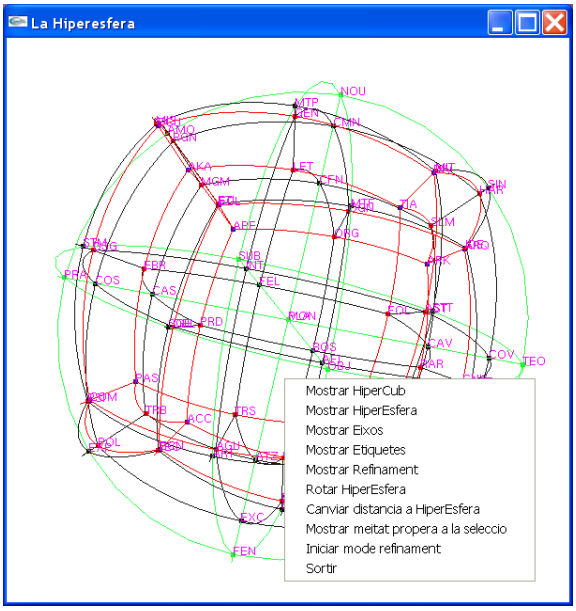


Figure 4. Options for the hyper-sphere

Mouse functions

The mouse buttons have some different functions:

Left button. This button selects a point or a section of the model. In order to achieve that, the mouse cursor is placed over the point (situated near the left inferior part of the first letter of the label) and presses the left button. The text console shows a message informing about the selected point.

Once the point has been selected, in case there is not refinement mode, the point and the label are highlighted in yellow, they appear in a bigger size and the chosen point is placed in the centre of the image.

In case refinement mode, the point will be coloured into purple and the label will temporarily appear in yellow and in a bigger size as mentioned. However, the image is not

reoriented to place the point in the centre: it stays in the same place waiting for the selection of new points or the end of the use of this refinement mode.

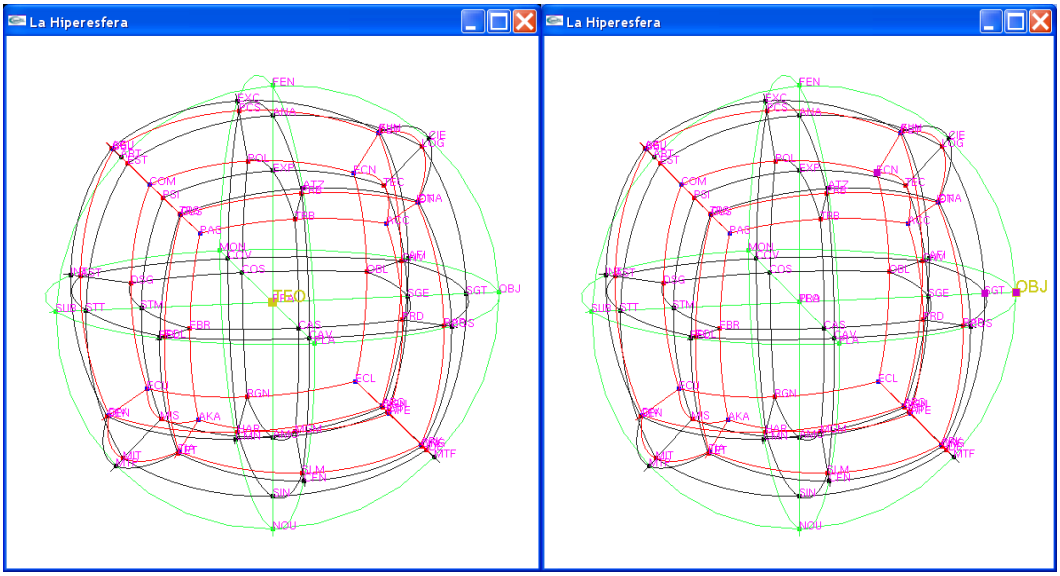


Figure 5. Hyper-sphere

Right button. On pressing the right button in any point of the graphic window, a pull-down menu appears. The elements of the menu are: Show hypercube, Show hyper-sphere, Show axes, Show labels, Show refinement, Rotate sphere, Change distance to hyper-sphere, Show half close to selection, Initiate refinement mode, Finish refinement mode and leave option. Next, there is a detail of each option.

Show hypercube. It is possible to visualise a hypercube or a four-dimension cube (formed by eight cubic faces) as some help to understand the hyper-sphere. The following figure is an example of the hypercube visualisation.

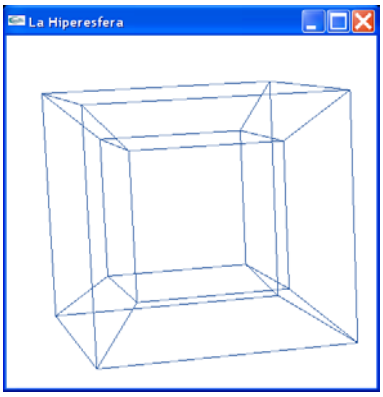


Figure 6. Hyper-cube visualisation

Show hyper-sphere. In this case, the menu allows either to visualise the hyper-sphere or not. 80 points and all the different relations that join these points form this hyper-sphere. It is visualised by default when the program is being initialised.

Show axes. It visualises or hides the reference axes, in the four dimensions: axis x in red, axis y in blue, axis z in blue and axis w in black. We imagine that all these four axes are perpendicular among them. The following shows the hypercube with the reference axes.

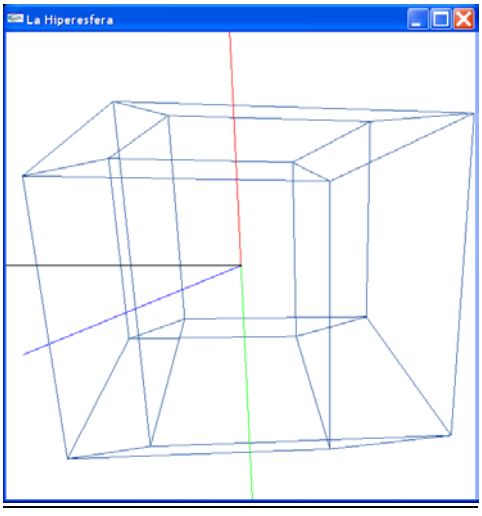


Figure 7. Hypercube with reference axes.

Show labels. It visualises or hides the labels associated with each of the 80 groups of policies. They are short forms formed by three letters of the concept related to the point.

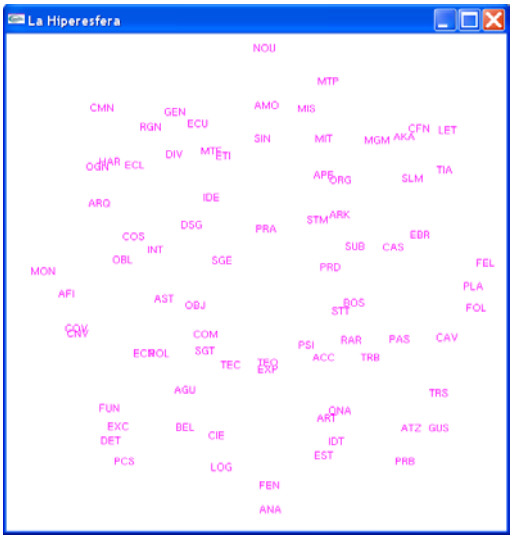


Figure 8. Labels associated with policies

Show refinement. It visualises or hides the refinement of the point network that the user has done. The new points are computed as an average point or barycentre of several selected points. They have not got a label and they are joined to the selected points by means of blue lines. Next there is an example where there is not visible the hyper-sphere, only the labels, just to remark the selection.

Rotate hyper-sphere. This menu activates or stops the automatic rotation of the hyper-sphere. The implemented rotation is in four dimensions and, therefore, it leaves two fix axes and the other two axes rotate.

Change distance to the hyper-sphere. This menu activates or stops the automatic movement of the camera to approach and move away periodically from the point in the centre of the scene. It is possible to activate this camera movement and the rotation one simultaneously.

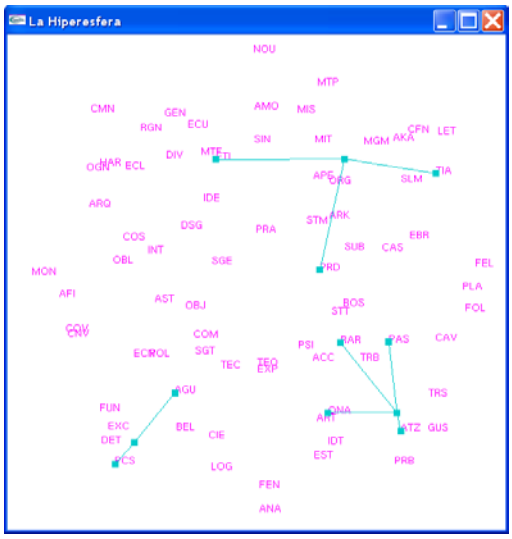


Figure 9. Labels of the hyper-sphere.

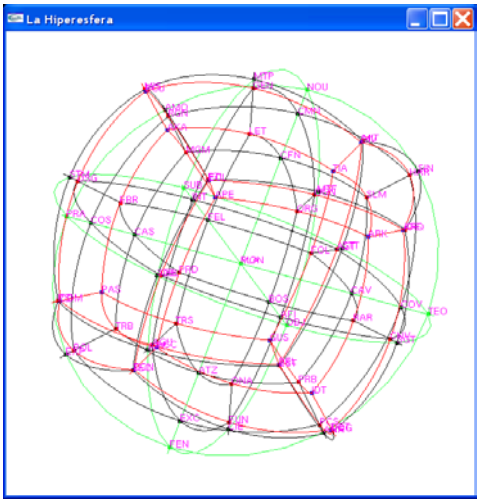


Figure 10. Rotate hyper-sphere

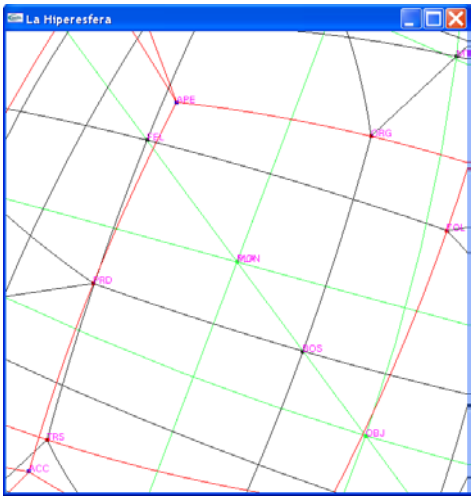


Figure 11. Change distance to the hyper-sphere

Show half close to selection. All the hyper-sphere is visualised at the same time by default, both the points closer to the camera and those that belong to the posterior part, which would be the hidden part if the hyper-sphere were opaque. In order to simulate the hiding of this posterior part, there is a menu that allows choosing between visualising all the hyper-sphere or just the visible half around the selected point. Therefore, a (hyper) hemisphere (with a

pole or the centre of the selected point) is only shown. The following picture shows all the hyper-sphere with the OBJ point selected; in the second one, only the hemisphere close to OBJ is shown with this point in the centre, and in the third one, the figure has been rotated in order to make visible a hemisphere.

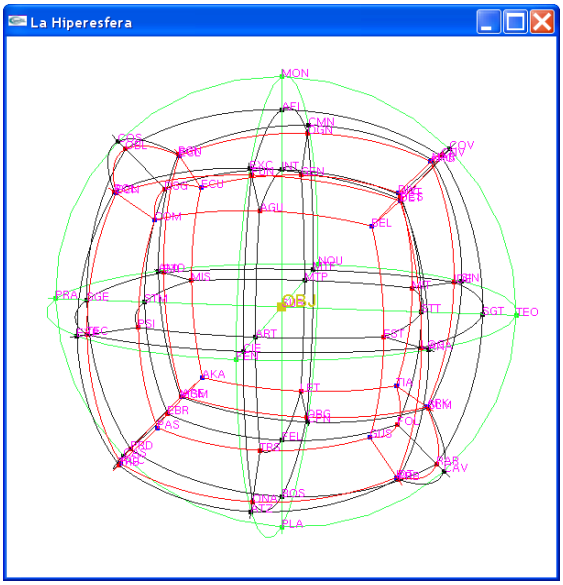


Figure 12. hyper-sphere with the OBJ point selected

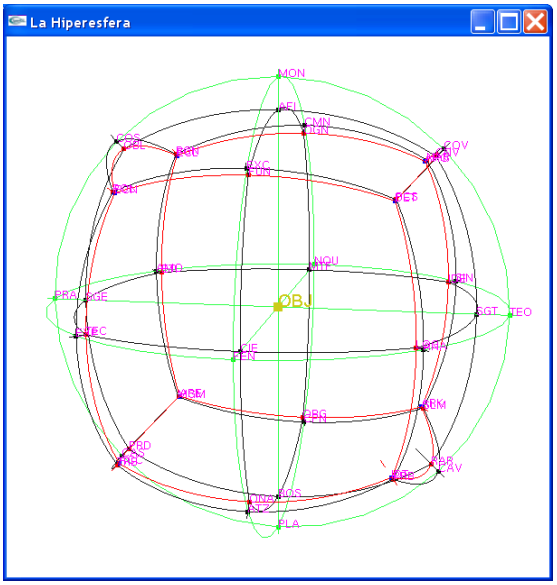


Figure 12. Hemisphere close to OBJ

II.5.1 Initiate the refinement mode / Finish the refinement mode

The selection of the *Initiate the refinement mode* from the menu, allows to add new points to the model, which will be the average point between two points, the barycentre of three points or the centre of the tetrahedron formed by four points. The refinement mode finishes with the option *Finish the refinement mode.*

The refinement mode selects points by means of the left button of the mouse, but always without placing it in the centre of the figure. Four points can be selected. Afterwards, the refinement mode is deactivated automatically and the new point is computed. To compute the barycentre of a triangle instead, once the three vertexes have been selected, it is

necessary to force the computation of the new point selecting the menu *Finish the refinement mode*. Then the coordinates of the new point are shown in the console.

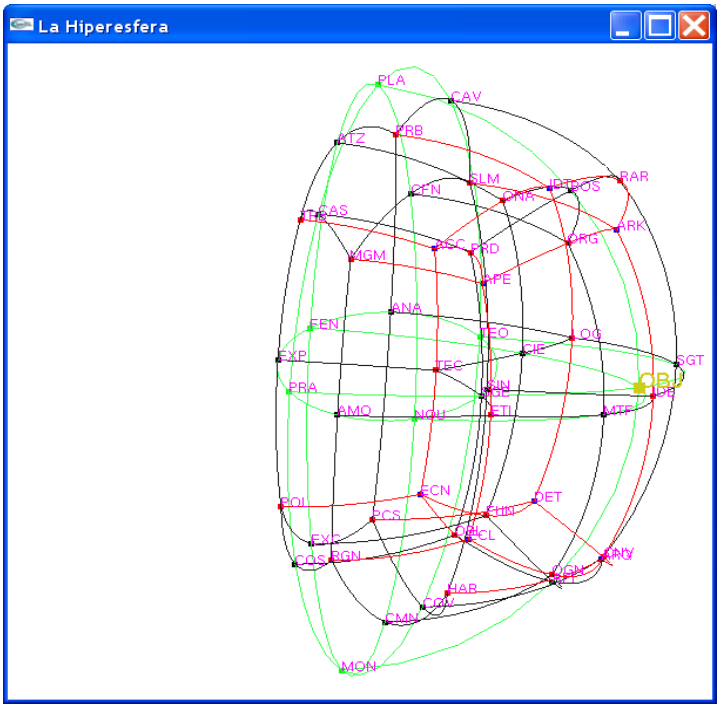


Figure 13. The figure has been rotated to make visible a hemisphere

Future research is related to correct a refinement and to select the new points coming from previous refinements. In the following example, the average point between PCS and AGU has been computed, the barycentre between ETI, TIA and PRD, and the centre of RAR, PAS, ONA and ATZ.

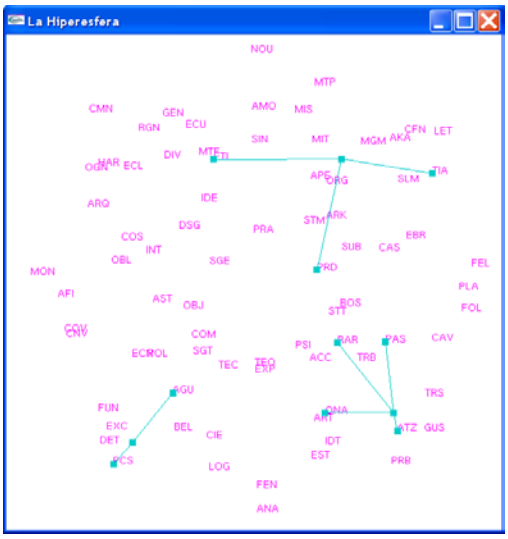
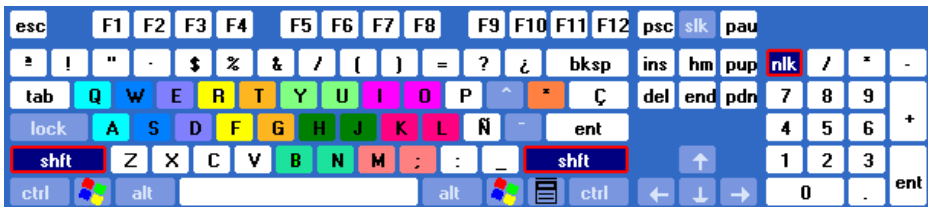


Figure 14. Refinements

Keyboard functions

The coloured keys have a function associated to them:



There are three groups of functionalities:

1. Leave: pressing the key (+) we end the execution of the program.
2. 3D camera movement: Keys (q), (a), (w), (s), (e), (d), (r), and (f) control the 3D camera position, which observes the models from any three-dimensional point of view.

Q	3D Rotation according to axis X
A	3D Rotation according to axis X (opposite direction)
W	3D Rotation according to axis Y
S	3D Rotation according to axis Y (opposite direction)
E	3D Rotation according to axis Z
D	3D Rotation according to axis Z (opposite direction)
R	Move forwards 3D Camera up to the centre of the scene (zoom approaching)
F	Move backwards 3D Camera from the centre of the scene (zoom moving away)

3. 4D camera movement: the rest of the keys control the 4D camera position, which changes the point from which the four-dimension model is projected into the three-dimension world. A rotation in four dimensions leaves two fix axes and rotates the other two. There are six possible rotations:

T	Move forwards 4D Camera up to the centre of the scene (zoom approaching) (equivalent to 3D)
G	Move backwards 4D Camera from the centre of the scene (zoom moving away) (equivalent to 3D)
Y	4D Rotation, rotating axes
U	4D Rotation, rotating axes (opposite direction)
H	4D Rotation, rotating axes
J	4D Rotation, rotating axes (opposite direction)
B	4D Rotation, rotating axes
N	4D Rotation, rotating axes (opposite direction)
I	4D Rotation, rotating axes
O	4D Rotation, rotating axes (opposite direction)
K	4D Rotation, rotating axes
L	4D Rotation, rotating axes (opposite direction)
M	4D Rotation, rotating axes
,	4D Rotation, rotating axes (opposite direction)

II.5.2 Conclusions about the application

The 4D module projection is carried out into the 3D one, which has been the most difficult process, and to make it useful to project any scene as defined in 4D, not just the hyper-sphere (in fact, it is possible to see the hypercube and also it is possible to add more figures).

The program about the user's interface requires more work, currently there is only a pull-down menu and some functions by means of the keyboard, but it gives complete access to all the functions and allows navigating over the scene interactively.

Some points can be selected in order to study them and refine the model just by adding some new points. However, these new points are not still a part of the model and in consequence, the refinements carried out cannot be filed.

II.6 Stages to follow in order to start the functioning of the Routing Distributed System based on PBMS

Next the different stages to start the functioning of the system into the GIRIN scenario are presented.

First stage: LDAP server start

The LDAP server is allocated on the PC with Linux. A secure connection (SSH) must be used to reach it. The server is usually active, that is the reason why it is not necessary to start it. However, it is important to know the mechanism used to start the service in case it fails because of the system fall. As a root user, the following command must be executed:

/usr/local/libexec/slapd -f /home/tebar/MAS.conf

Second stage: BBDD Oracle start

BBDD server start. The ORACLE server is allocated on the 10 UltraSparc called **c3gestor.upc.es**. It happens the same that in the case just above mentioned, that is to say, it is always active as a default mechanism and, even when the system falls, it restarts at the same time as the operative system does. In any case, the commands that must be used both for the start and the stop of the server are the following: (They should be executed by the oracle user).

/u01/app/oracle/product/8.0.4/bin/dbstart

/u01/app/oracle/product/8.0.4/bin/dbshut

ORACLE listener start. It is necessary to start this process in charge of listening the requests at the BBDD Oracle through the port 1521 and sending the queries result through the created socket. Unlike the aspects above mentioned, this process does not start automatically with the operative system. This is the reason why it's functioning must be checked periodically or in case some problems emerge in the BBDD Oracle requests. The lsnrctl (oracle user) is the command that must be used and the start listener plus exit in the lnrctl subsystem.

```
$ lsnrctl
LSNRCTL for Solaris: Version 8.0.6.0.0 - Production on 03-JUN-01 17:22:04
(c) Copyright 1999 Oracle Corporation.  All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> start listener
Starting /u01/app/oracle/product/8.0.4/bin/tnslsnr: please wait...
TNSLSNR for Solaris: Version 8.0.6.0.0 - Production
System parameter file is /u01/app/oracle/product/8.0.4/network/admin/listener.ora
Log messages written to /u01/app/oracle/product/8.0.4/network/log/listener.log
Listening on: (ADDRESS=(PROTOCOL=ipc)(DEV=8)(KEY=c3ge))
Listening on: (ADDRESS=(PROTOCOL=ipc)(DEV=12)(KEY=PNPKEY))
Listening on: (ADDRESS=(PROTOCOL=tcp)(DEV=14)(HOST=147.83.39.123)(PORT=1521))

Connecting to (ADDRESS=(PROTOCOL=IPC)(KEY=c3ge))
STATUS of the LISTENER
```

Alias	Listener
Version	TNSLSNR for Solaris: Version 8.0.6.0.0 – Production
Start Date	03-JUN-01 17:22:23
Uptime	0 days 0 hr. 0 min. 0 sec
Trace Level	off
	OFF
Security	OFF
SNMP	/u01/app/oracle/product/8.0.4/network/admin/listener.ora
Listener Parameter File	/u01/app/oracle/product/8.0.4/network/log/listener.log
Listener Log File	

Table 10. ORACLE listener start

c3ge	has 1 service handler(s)
extproc	has 1 service handler(s)

Table 11. Service summary

The command completed successfully
LSNRCTL> exit

An analogue proceeding with the stop listener subcommand would be used to stop the listener.

COSnaming service start

Any user can (and must) start the CORBA Name System. The information referring to all and every of the objects created during the system functioning is kept in it: location, status, interface, etc. Provided the fact that CORBA is a distributed computation system, the COSNaming Service could be at any network computer. In our case and due to simplicity aspects, the chosen server has been the UltraSparc 10 c3gestor.upc.es again. In order to start it, the port 1050 port is used, which was selected arbitrarily, and the command tnameserv -ORBInitialPort 1050 &.

Start of the management application server

The application server is also allocated on c3gestor.upc.es. Due to the fact that the application has been developed in java, its execution implies the creation of a java virtual machine that executes the MAS class. The functioning directory is /home/tebar/idl and the used command is java MAS.

Router start

The machine in charge of act as a router in this first stage is the UltraSparc 5 girin-0.upc.es. The SecondTier.class must be started in it. The functioning directory is /home/tebar/idl and the used command is java SecondTier.

From this moment on, we can say that the system is completely operative and ready to receive new requests from the different clients.

Client program execution

Finally, the user program is started from the PClient.class. It can be executed in any machine of the network, but it must always consider the Java Runtime Environment 1.3.0_02 or superior. The functioning directory is /home/tebar/Client and the used command is java PClient.

References

[Barba02] Antonio Barba, Joel García, Lluis M. Xirinacs..*Manual para representaciones geometricas de politicas*. Documento interno Dep. ENTEL. UPC. 2002.

[Demichelis02] C. Demichelis, P. Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). IETF Request For Comments (RFC) 3393. November 2002.

[Grossman02] D. Grossman New Terminology and Clarifications for Diffserv. IETF Request For Comments (RFC) 3260 April 2002.

[Zienckiewicz] O.C. Zienckiewicz - R. L. Taylor. *El Método de los Elementos Finitos*. Ed. McGraw Hill / CIMNE

[Reyes00-telecom] Angélica Reyes, Antoni Barba *Desarrollo de una aplicación de Gestión de encaminamiento en Internet2* Telecom 2000. Madrid España.

[Reyes00-URSI] Angélica Reyes, Antoni Barba Gestión de encaminamiento basado en restricciones de Calidad de Servicio en Internet2 Union Radio-Scientifique Internacionale (URSI 2000). Zaragoza España.