



**Escola Politècnica Superior
de Castelldefels**

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FIN DE CARRERA

TÍTULO DEL TFC: Implementación de un sistema de direccionamiento intencional

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Antoni Carenys Roca

DIRECTOR: Juan López Rubio

FECHA: 4 de junio de 2009

Título: Implementación de un sistema de direccionamiento intencional

Autor: Antoni Carenys Roca

Director: Juan López Rubio

Fecha: 4 de junio de 2009

Resumen

El objetivo principal de este proyecto, es el diseño y posterior implementación de un sistema de direccionamiento, basado en direcciones o nombres intencionales. Este sistema permite que un usuario pueda buscar mediante un identificador, los servicios con las características que le interesan, describiéndolas el mismo.

Estos servicios, primero se han anunciado por la red, autodescribiéndose mediante un identificador con un formato en texto basado en pares de atributos y valores, que no están previamente definidos. Estos conjuntos de atributos y valores, están agrupados mediante claudators, que indican si un par está relacionado con otro par: *[país = España [ciudad = Barcelona]]*.

Una vez este identificador llega al servidor, este lo separa para extraer los pares de atributos y valores, y guarda la información en un árbol de información que va autogenerándose solo. En el último valor de cada rama en las que se ha guardado el identificador, hay una lista en la que se agrega la dirección IP de ese servicio y de los otros servicios que tengan en común esa característica descrita en el identificador. Cada IP servirá de índice para otra base de datos con otra información extra del servicio.

A partir de que el servidor contiene información de los servicios anunciados, los clientes pueden hacer búsquedas remotamente. Estas búsquedas las personaliza cada cliente a su manera, según las características del servicio o servicios que quiere encontrar. Para realizarlas, los clientes crearán un identificador con la misma estructura que los identificadores que envían los servicios para anunciarse. Además, los clientes también pueden buscar desde una IP, para saber todas sus características en formato identificador.

Para gestionar el servidor, se han implementado una serie de herramientas de gestión, que permiten administrarlo remotamente y obtener información de los errores que aparecen.

Se ha apostado por este sistema, ya que los actuales sistemas de nombres, no ofrecen mucha personalización, y no permiten hacer búsquedas intencionales, indicando exactamente las características de lo que se quiere buscar. Gracias a este sistema, se ha conseguido un nuevo sistema de direccionamiento que consume pocos recursos, y es altamente personalizable en cuanto a la forma de guardar información de los anunciantes.

Title: Implementation of an intentional addressing system

Author: Antoni Carenys Roca

Director: Juan López Rubio

Date: June, 4th 2009

Overview

The main objective of this project is the development and subsequent implementation of an addressing system, based on intentional addresses or intentional names. This system allows to the users, to search by an identifier, services with the features that interest to them, describing this features, themselves.

These services were first announced by the network, describing itself by an identifier with a text format, based on pairs of attributes and values that are not previously defined. These sets of attributes and values, are grouped by square brackets that indicate if a pair is related to another pair: *[country = Spain [city = Barcelona]]*.

Once this identifier reaches the server, it separates the identifier for extracting the pairs of attributes and values, and stores the information in an information tree that is generated itself. In the last value of each branch in which the server saved the identifier, it add in a list, the IP address of that service and the other services that are common in this feature described in the identifier. Each IP will serve index database for other extra information service.

When the server contains information of the services announced previously, users can search remotely. These searches are customized by each client depending on the type of service or services that they want find. To perform these searches, customers create an identifier with the same structure of the identifiers delivered by services to advertise itself. In addition, customers can also find from an IP, to know all its features in identifier format.

To manage the server, have implemented some management tools that allow managing the server remotely and details the errors that appear in the server.

We have chosen for this system, because the current systems names, do not offer much customization, and they do not do intentional searches, indicating the exact characteristics of whatever you want to search. With this system, we have introduced a new addressing system that consumes fewer resources and is highly customizable in how to save information from advertisers.

ÍNDICE

INTRODUCCIÓN	1
CAPÍTULO 1. EL SISTEMA	2
1.1. Los sistemas de nombres	2
1.1.1. Sistema de nombres de domino (DNS).....	2
1.1.2. Sistema intencional	3
1.2. Diseño del sistema intencional.....	3
1.2.1. Identificador	3
1.2.2. Generación del árbol	5
1.2.3. Información adicional.....	9
1.2.4. Encontrar la información.....	11
CAPÍTULO 2. IMPLEMENTACIÓN	15
2.1. Servidor	15
2.1.1. Funciones centrales	15
2.1.2. Funciones de red.....	19
2.2. Clientes.....	21
2.2.1. Cliente de anuncio.....	21
2.2.2. Cliente de búsqueda.....	22
2.2.3. Cliente de gestión.....	24
CAPÍTULO 3. CASO PRÁCTICO.....	29
3.1. Empresa textil	29
CAPÍTULO 4. PRUEBAS	40
4.1. Uso de red	40
4.2. Velocidad.....	42
4.2.1 Según ordenador.....	42
4.2.2 Según tamaño del árbol	45
CAPÍTULO 5. IMPACTO AMBIENTAL	50
CAPÍTULO 6. CONCLUSIONES.....	51
BIBLIOGRAFIA	52
ANEXO A. Ejemplo de archivo donde el servidor guarda el árbol.....	55

INTRODUCCIÓN

Hoy en día, los sistemas de nombres que existen, no nos dejan hacer búsquedas complejas y detalladas acerca de lo que queremos buscar, sino que se limitan a convertir un nombre legible por un humano, a una dirección IP. Un cliente que quiera buscar algo, no tiene poder para personalizar su búsqueda, sino que se ve envuelto dentro de un tipo de nombre predefinido en el que además de no dar información, no deja cambiarlo para la búsqueda.

Debido a esto, con este trabajo, se ha implementado un sistema intencional de búsquedas de direcciones. Un sistema intencional nos tiene que permitir describir mediante el nombre, lo que queremos encontrar, indicando sus características, y a la vez, el servidor que contenga la información, nos tendrá que devolver todo lo que se corresponda con lo que hemos indicado en las búsquedas que se realizan.

Pero para poder hacer esto, el servidor tiene que guardar la información de una manera que le permita rápidamente hacer estas búsquedas, que a veces serán simples, y otras veces mucho más complejas. También se tiene que llegar a diseñar una estructura para los nombres que utilizan los servicios para anunciarse y los que utilizarán los clientes para realizar las búsquedas. Estos nombres tienen que tener un gran margen de personalización, para permitir describir cualquier información.

En el primer capítulo, se desarrolla y describe las características que ha de tener el sistema que se va a implementar, así como la mejor manera de guardar toda la información en el servidor, de la forma que nos permita hacer búsquedas complejas.

Una vez se entra en el segundo capítulo, se describe la implementación que se ha hecho de las ideas descritas en el anterior capítulo. Se muestra las distintas aplicaciones programadas, tanto en el lado del servidor, como en los lados de los clientes, y se explican cada una de las funciones que pueden realizar.

Para que todo se entienda mejor, en el tercer capítulo se muestra una posible aplicación del sistema implementado, en una tienda. El ejemplo muestra paso a paso de forma visual todos los pasos que se van produciendo desde que se anuncian los servicios, hasta que los clientes quieren buscar algo.

El cuarto capítulo, describe diversas pruebas que se han realizado, que pueden servir para saber la carga que puede soportar el sistema, y la velocidad de búsqueda que puede haber en el servidor, a medida que va aumentando la información guardada.

Finalmente se analizan y debaten las conclusiones a las que se han llegado desarrollando e implementando el sistema intencional, a partir de sus puntos fuertes y los débiles.

CAPÍTULO 1. EL SISTEMA

1.1. *Los sistemas de nombres*

En la actualidad, los sistemas de nombres, como los DNS (Domain Name System), no nos proporcionan información detallada de lo que estamos buscando o de del servicio que queremos utilizar, ya que nos proporcionan la dirección IP para poder saber su localización.

Además, no nos permiten hacer búsquedas más complejas indicando por ejemplo las características o funcionalidades que nos puede proporcionar el servicio que queremos buscar concretamente.

Estos sistemas conocidos actualmente, únicamente traducen un nombre que puede ser leído y entendido por un humano, a su localización en la red, es decir, su dirección IP. Gracias a esto se hacen unas conversiones muy simples, pero que son tan simples que no dejan un gran margen de personalización.

1.1.1. **Sistema de nombres de domino (DNS)**

Los DNS, generalmente sirven para hacer peticiones de la localización (su dirección IP), de un cierto nombre de dominio que le indicamos, pero ni el nombre de dominio ni la dirección IP, nos indican ninguna característica de ese servicio o lugar.

Su uso más común está en los navegadores, los cuales por ejemplo pueden preguntar qué IP tiene la página “*www.google.es*”, y el servidor de nombres les respondería “*209.85.229.104*”. De esta forma, el navegador puede acceder al lugar de la red donde está la página web, y mostrarla al usuario.

Como se ha visto en el ejemplo, con los sistemas DNS, buscamos algo que no se sabe lo que es, ni que características tiene, y que desde el nombre, aunque lo pueda leer un humano, este casi no nos proporciona ningún tipo de información, y además no podemos restringir la búsqueda en el caso de que quisiéramos hacer una búsqueda adecuada a nuestros gustos.

Vemos que un DNS no nos deja mucho margen, ya que no ha sido creado para nada más que ubicar un nombre concreto, y por esto es necesario un nuevo sistema para poder hacer conversiones más complejas de nombres más complejos y detallados, que puedan generar tanto el usuario que quiere buscar, como el servicio que se quiere anunciar

1.1.2. Sistema intencional

Un sistema intencional, a diferencia de los otros sistemas de nombres, nos permiten por un lado conocer las características de cierto servicio solo con el nombre, y por otro lado, poder hacer búsquedas de un servicio, creando un nombre que describa lo que nos interesa encontrar o acceder.

Un ejemplo de DNS intencional, sería que mediante el nombre de dominio, le pidiésemos las webs que tratan de informática y que además fuesen españolas, y el servidor nos devolviera todas las direcciones IP de lo que se corresponde a nuestra búsqueda.

1.2. *Diseño del sistema intencional*

Este sistema intencional, está basado en el artículo *“The design and implementation of an intentional naming system”* [2]. A partir de este artículo, adaptaremos a nuestras necesidades un sistema intencional, para posteriormente proceder a implementarlo.

El sistema, permite que los servicios se puedan anunciar a los servidores, explicando de una forma muy simple sus características, por las cuales más tarde un cliente le puede buscar.

Para poder realizar esto, se diseña un nombre identificador que los servicios utilizaran para anunciarse, y una base de datos en forma de árbol en el servidor, para guardar esta información del nombre identificador. Además, para cada servicio, también se puede guardar información extra por la que no se quiere que se le busque, como tiempo de expiración, la cual sea guardada aparte.

1.2.1. Identificador

El identificador o nombre identificador, es una cadena de texto, con la que los servicios se anuncian, indicando así información principal como puede ser su localización, accesibilidad, velocidad, etc. La información que contiene el identificador, la escoge el servicio, y no tiene porque siempre contener el mismo tipo de información en cada servicio que se anuncia, generando de esta forma un sistema no regulado por normas ni idiomas.

El identificador, está formado por un conjunto de parejas de atributos y valores, envueltas por corchetes, en las que el atributo está separado del valor por un signo igual:

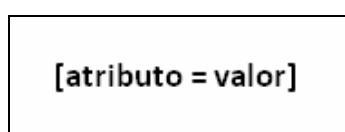


Fig. 1.1 *Formación simple del identificador*

No hay un listado predefinido de atributos y valores, sino que cada servicio, escoge los que cree que serían más útiles para darse a conocer, para así también hacer un diseño en el que los clientes y servicios, tienen libertad y se genera una autoconfiguración.

Algunos ejemplos de atributos y valores, podrían ser los siguientes:

```
[ciudad = barcelona]
[edificio = torre agbar]
  [planta = 4]
[accesibilidad = privada]
[servicio = impresora]
```

Fig. 1.2 *Ejemplos varios de identificadores simples*

Como se observa en los ejemplos anteriores, hay algunos pares, que están muy relacionados con otros, o que tendrían que pertenecer a otros, ya que uno estaría dentro del otro, como es el caso de “*planta*” que pertenece a “*edificio*”, o el de este mismo “*edificio*” que pertenece a “*ciudad*”.

En este caso, para formar el nombre identificador, se ha ideado una lógica muy simple, pero con muy buen resultado, y es la de poner la pareja de atributos y valores, dentro de la otra pareja a la que corresponde, entre el valor y el corchete que indica final de pareja. De esta forma, se obtiene un nombre simple, y a la vez fácil de entender tanto a simple vista por un humano, como por un programa de ordenador.

Visualmente, siguiendo la lógica mencionada, la cadena se iría formando de esta forma, incluyendo un par dentro del otro:

```
[ciudad = barcelona]
[ciudad = barcelona [edificio = torre agbar]]
[ciudad = barcelona [edificio = torre agbar [planta = 4]]]
```

Fig. 1.3 *Identificadores relacionados entre si*

Siguiendo esta estructura, se iría generando un nombre identificador que cada vez contendrá más información, y describirá tanto como es el servicio a donde está situado por ejemplo, llegando a generar un nombre identificador más complejo como el siguiente:

```
[ciudad = barcelona [edificio = torre agbar [planta = 4 [sala = b]]]]  
[servicio = impresora [color = si [ppm = 20]]]  
[accesibilidad = privada [grupo = rrhh  
                        [grupo = it]]]
```

Fig. 1.4 *Identificador complejo*

Como se observa, en el nombre identificador anterior, se anuncia una impresora a color que imprime a 20 ppm, la cual está situada en la sala b de la cuarta planta del edificio Agbar de Barcelona, y que su accesibilidad es privada pudiendo solo imprimir en ella los usuarios de RRHH y IT.

Es verdad que a simple vista, sin poner los pares uno dentro del otro, un humano también podría conocer esta relación de información a simple vista, pero como la información la tiene que tratar un programa informático, esta es la mejor forma. De esta manera, también se evitan malentendidos de lectura y comprensión.

1.2.2. Generación del árbol

Una vez el nombre identificador llega al servidor, la información descrita en este identificador, tiene que poder guardarse de una forma que permita ordenar la información sin complicación, y libremente.

La forma de guardarlo, nos tiene que permitir acceder a los datos fácilmente, y hacer búsquedas rápidas, aunque las búsquedas sean más complejas como podría ser buscar servicios de varios tipos concretos.

El sistema ideado, es un sistema en el que la información se estructura en forma de árbol. Este árbol nace en un punto principal llamado “root”, desde el cual salen el primer nivel de atributos.

El árbol de información, empieza desde cero, y sin ningún tipo atributos o valores predefinidos. De cada uno de estos atributos, salen el valor o valores que contiene, y de cada valor, salen los siguientes atributos que se anuncian relacionados al anterior.

Siguiendo esto, cuando al servidor le llega el nombre identificador que hemos confeccionado anteriormente, generaría un árbol con el siguiente aspecto:

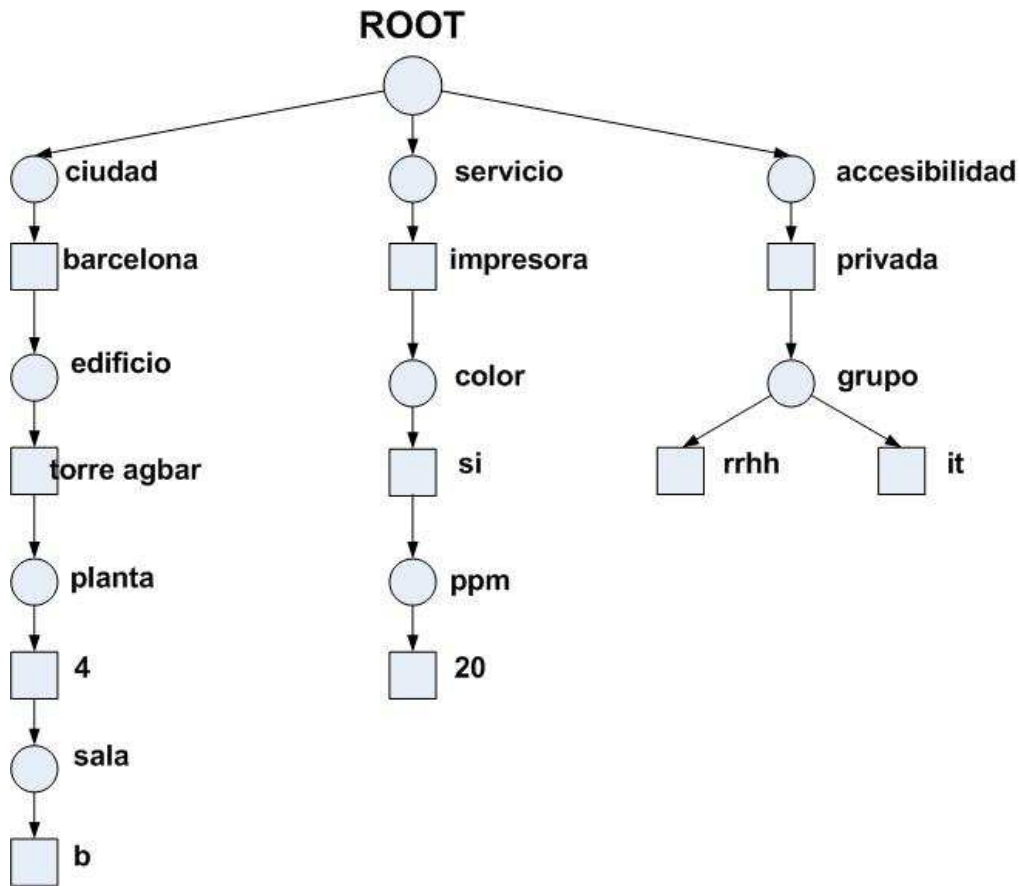


Fig. 1.5 Árbol simple

A medida que irá recibiendo identificadores de cada uno de los servicios que se anuncien, el árbol se irá autogenerando, y esta es la particularidad que hará al servidor muy flexible al no tener que seguir unas pautas de atributos, cosa que también hace que el servicio tenga mucha más flexibilidad para indicar la información que quiere dar a conocer.

Cuando el servidor que contiene el árbol generado anteriormente en el ejemplo recibe un nuevo anuncio, lo añadirá al árbol que hay creado, generando las ramas nuevas necesarias.

[ciudad = madrid [plaza = cibeles]]
[servicio = impresora [color = no [ppm = 30]]]
[accesibilidad = publica]

Fig. 1.6 Nuevo identificador

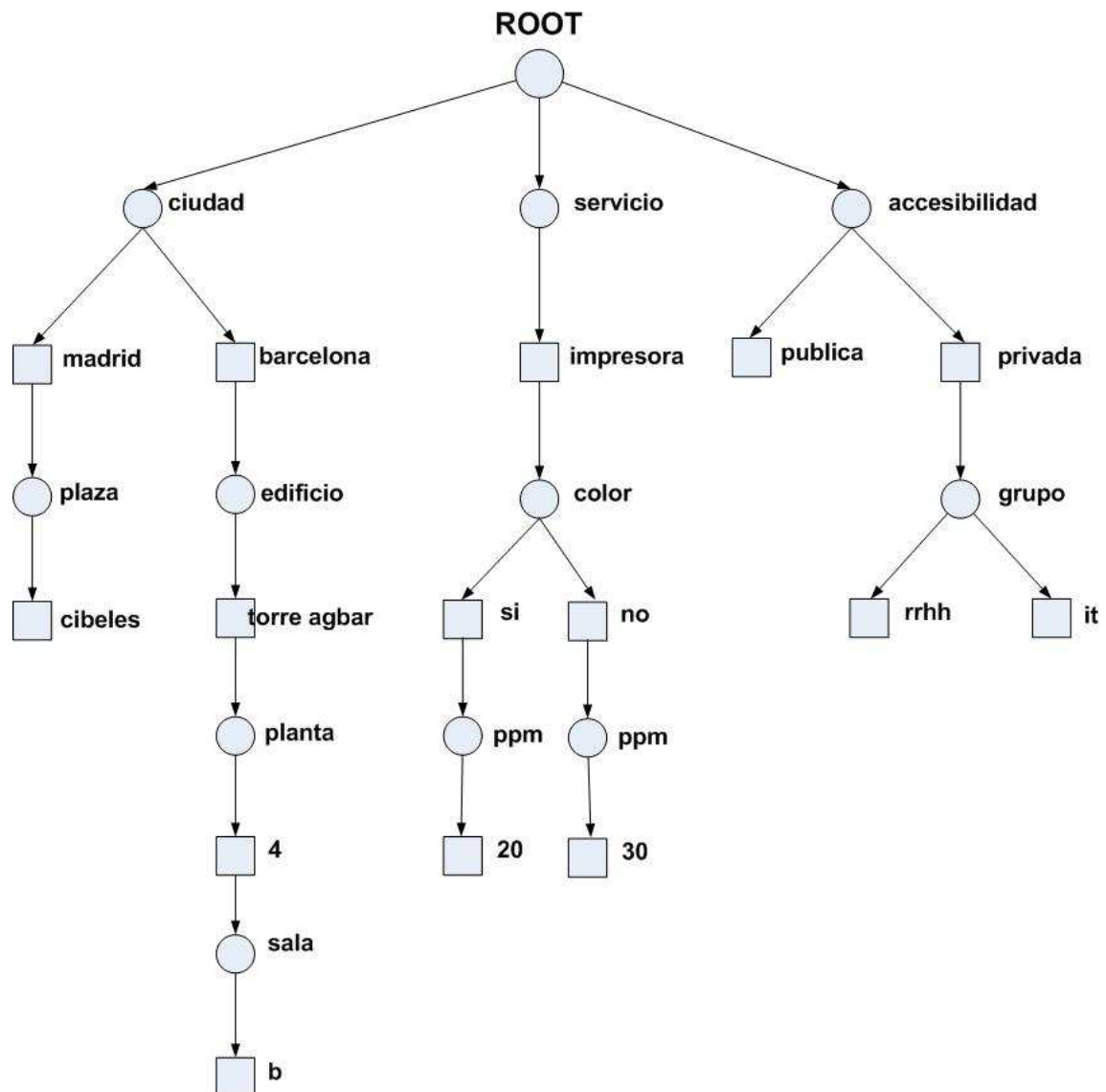


Fig. 1.7 *Árbol anterior con el nuevo identificador*

Vemos que al guardar el nuevo identificador que ha llegado, los atributos y valores que ya existían, se mantienen, y los nuevos que son comunes con el anterior, no se han añadido de forma duplicada.

En cambio, todos los atributos y valores nuevos que no existían anteriormente, se han agregado en forma de nuevas ramas que salen de los atributos o valores que ya había.

Nuevamente, si al árbol anterior, que ya contiene dos identificadores guardados, le llegan nuevos anuncios con identificadores, y estos se le van añadiendo, este árbol llegaría a tener una estructura mucho más completa y detallada, como la de la siguiente imagen:

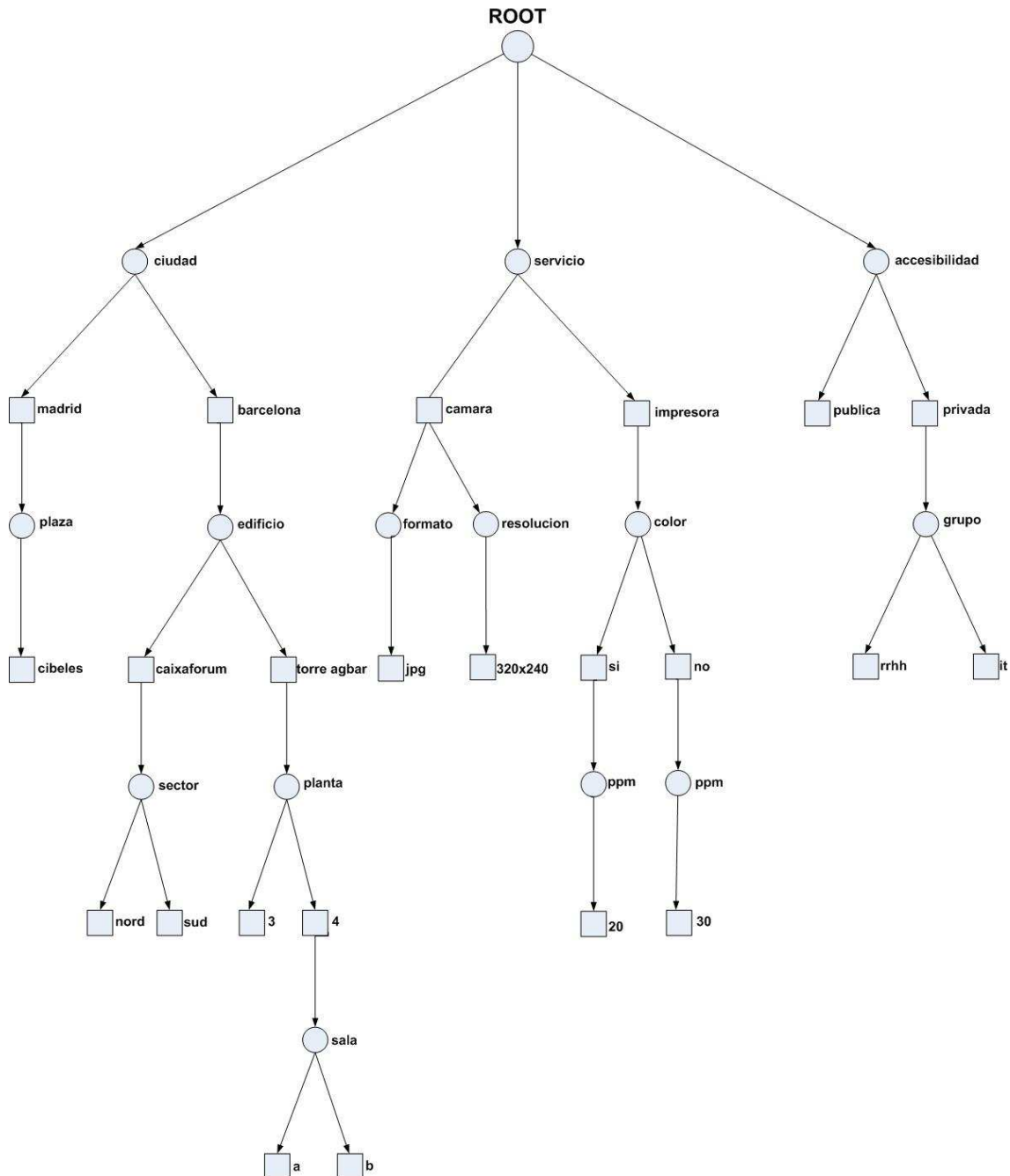


Fig. 1.8 *Árbol complejo*

1.2.3. Información adicional

Junto con el identificador, también se envía otra información extra o nameinfo que puede ser útil para el servidor, como el tiempo de expiración de la validez de ese servicio, o otra información que puede informar de otras características del servicio, pero que no se incluyen en el identificador, ya que no hace falta o no interesa que alguien encuentre dicho servicio buscando esa información.

Esta información extra del nombre o servicio, se guarda en otra base de datos dentro del servidor, pero relacionada a su vez con el árbol de información.

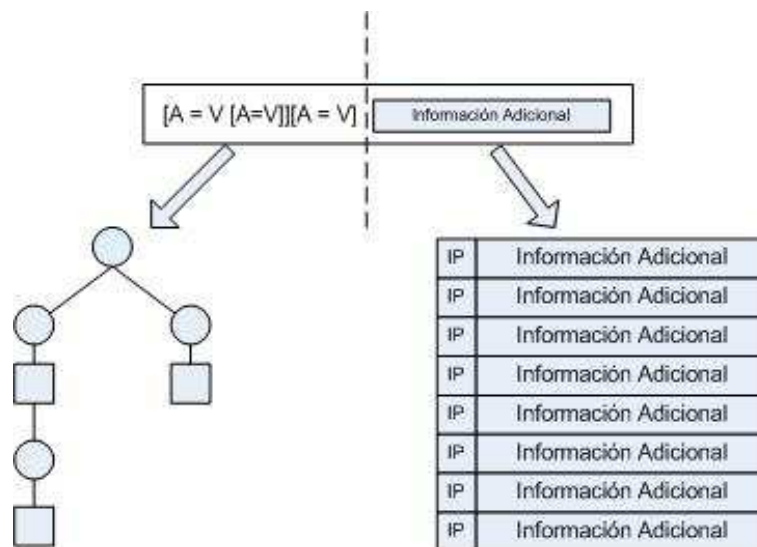


Fig. 1.9 Árbol e información extra

Cuando llega un nombre con la información extra, se añade la información extra a la segunda base de datos, y se guarda el identificador en el árbol, poniendo al final de cada rama generada en el árbol, una referencia que indica a que entrada de la base de datos corresponde.

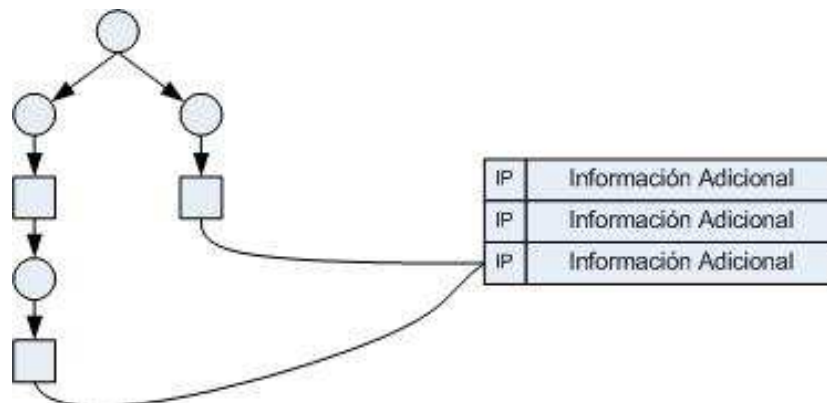


Fig. 1.10 Relación del árbol con la información extra

Siguiendo los ejemplos seguidos para crear el árbol, el árbol que contiene los dos identificadores que le han llegado y su relación con la información extra o nameinfo sería tal que así:

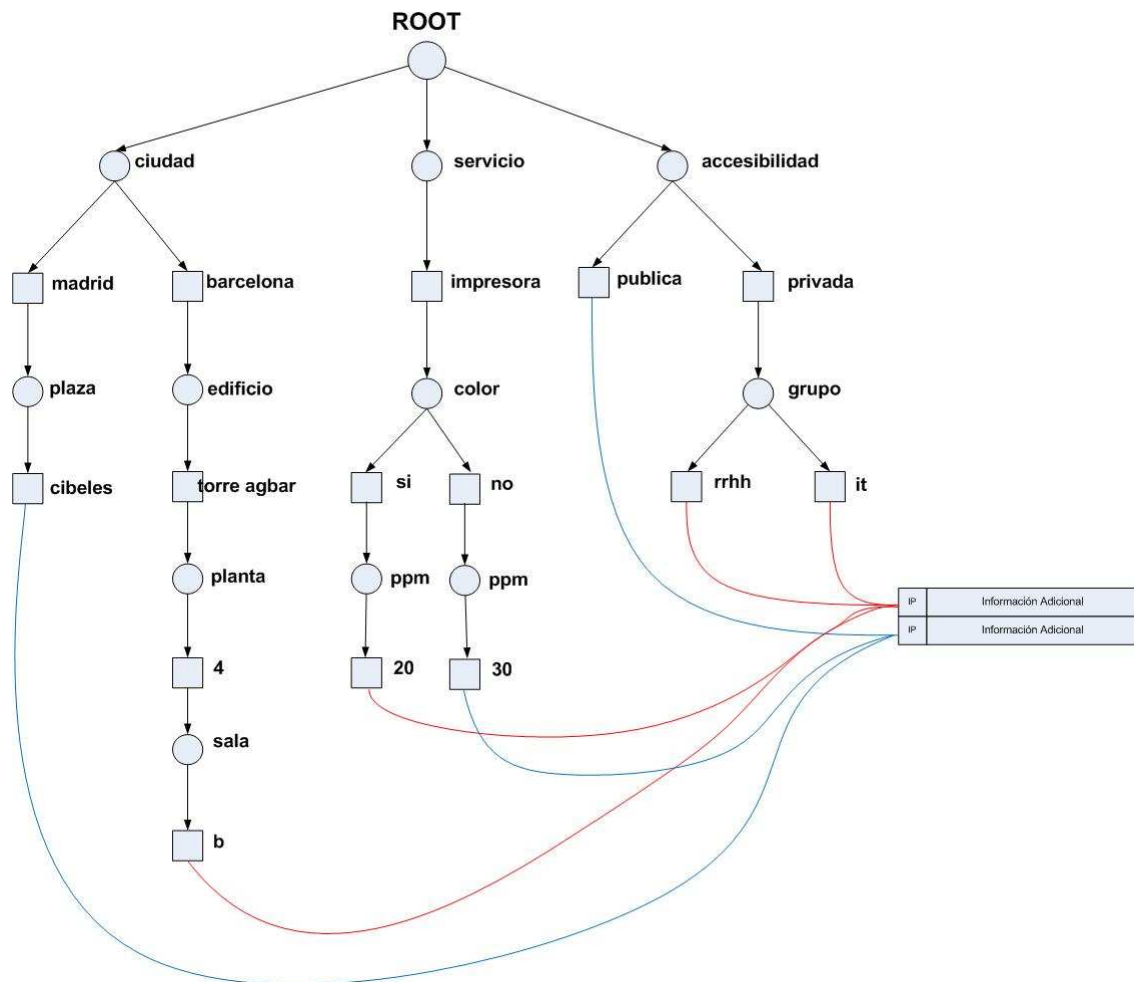


Fig. 1.11 Árbol y su relación con la información extra o nameinfo

Cada último valor de una rama que describe a un servicio, está relacionado mediante su dirección IP, con su posición en la base de datos que contiene la información extra.

Al estar solo relacionado el último valor de cada rama que describe a un servicio, como es lógico, el servicio entiende que los valores de allí hacia arriba también describen a ese servicio. Esto servirá para cuando un cliente quiera hacer búsquedas como por ejemplo indicando que quiere saber lo que hay en la ciudad de Barcelona, sin tener en cuenta lo que haya más hacia abajo.

Finalmente, uniendo a gran escala todo lo mencionado, la diversa información respecto a los servicios que se queda guardada en el servidor, tendría esta estructura:

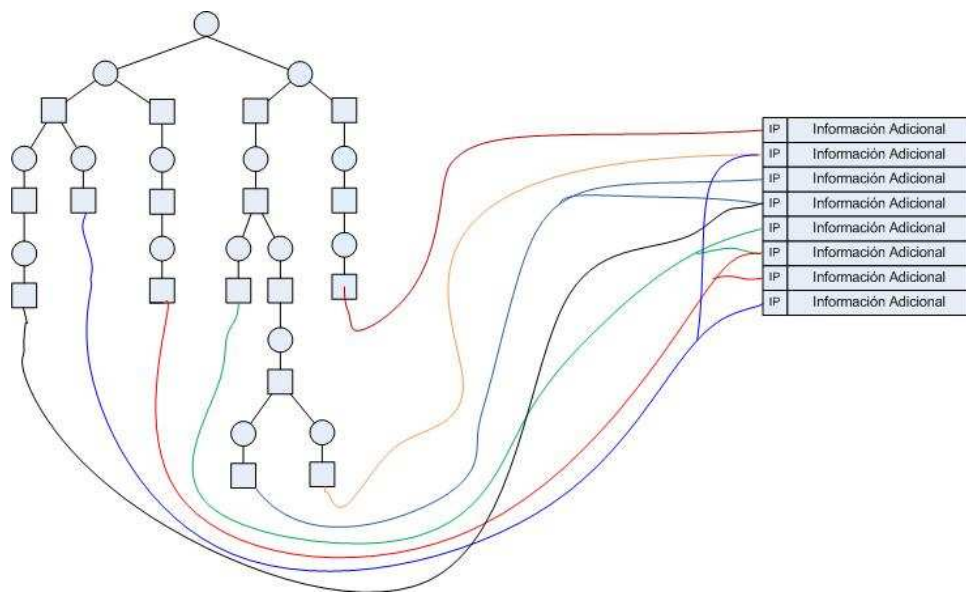


Fig. 1.12 *Árbol complejo y su relación con la información extra*

Una vez guardada toda la información en el árbol y en la base de datos de información adicional, el siguiente paso es poder hacer búsquedas en esa información, ya que es la utilidad de haberlo guardado con esta estructura.

1.2.4 Encontrar la información

Gracias a este sistema, el cliente podrá buscar mediante el envío de un nombre identificador que generará siguiendo la misma lógica que utilizan los servicios para anunciarse ($[A=V[A=V]]$), el tipo de servicio que le interesa, obteniendo todos los resultados que se adecuen a sus criterios de búsqueda.

De esta forma el cliente que quiere buscar, podrá autogenerar un nombre identificador para buscar lo que le interesa con total libertad.

Pongamos un primer caso, que a un cliente, le interesa saber que impresoras hay en la sala b de la planta 4 de la torre Agbar de Barcelona. Entonces este enviaría un identificador así:

[ciudad = barcelona [edificio = torre agbar [planta = 4[sala = b]]]]

Fig. 1.13 *Identificador*

Una vez llegase al servidor, este, iría recorriendo el árbol, en búsqueda de resultados, siguiendo los atributos y valores que se le describen en el identificador que ha recibido.

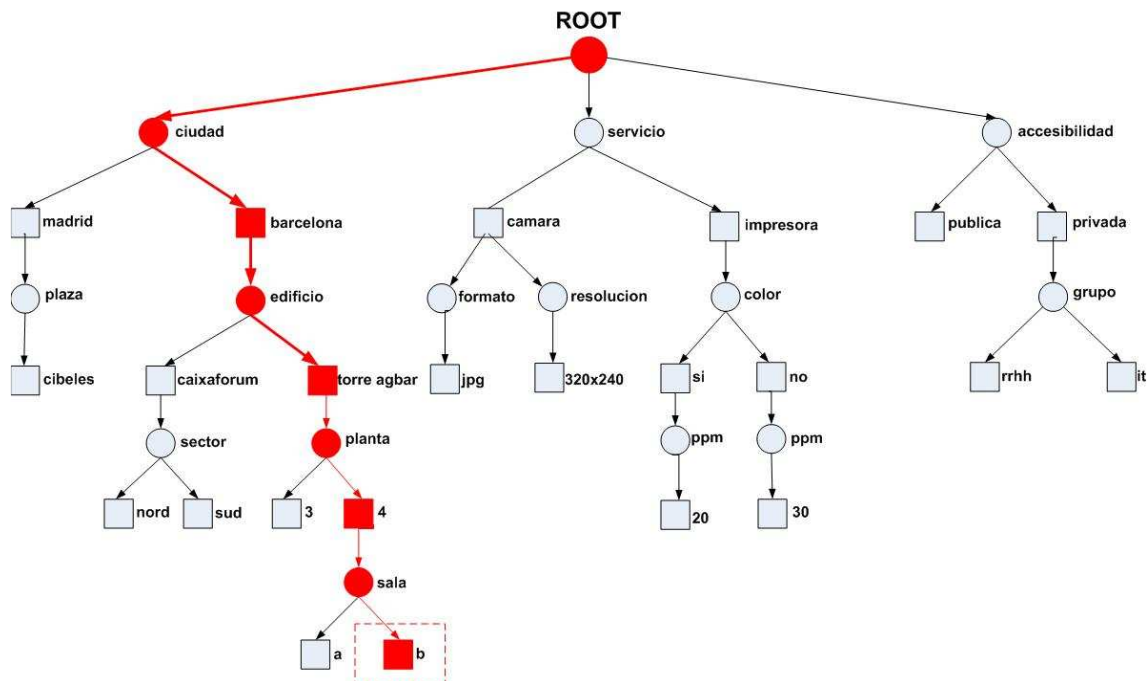


Fig. 1.14 *Búsqueda por el árbol*

El servidor va mirando si existen los atributos y valores mencionados, y en el caso de que existieran, iría bajando por las ramas del árbol de los atributos y valores nombrados.

Al final, como se han encontrado todos los atributos y valores, y organizados de la forma descrita en el identificador enviado para la búsqueda, el servidor devolvería todos los resultados en los que hay referencia en el valor final “b”.

En otro supuesto caso de que el cliente quisiera hacer una búsqueda un poco menos restrictiva, podría preguntar esta vez solo por todo lo que hay en la torre Agbar de la ciudad de Barcelona, sin tener en cuenta en que piso o sala está, mediante el siguiente identificador:

[ciudad = barcelona [edificio = torre agbar]]

Fig. 1.15 *Identificador*

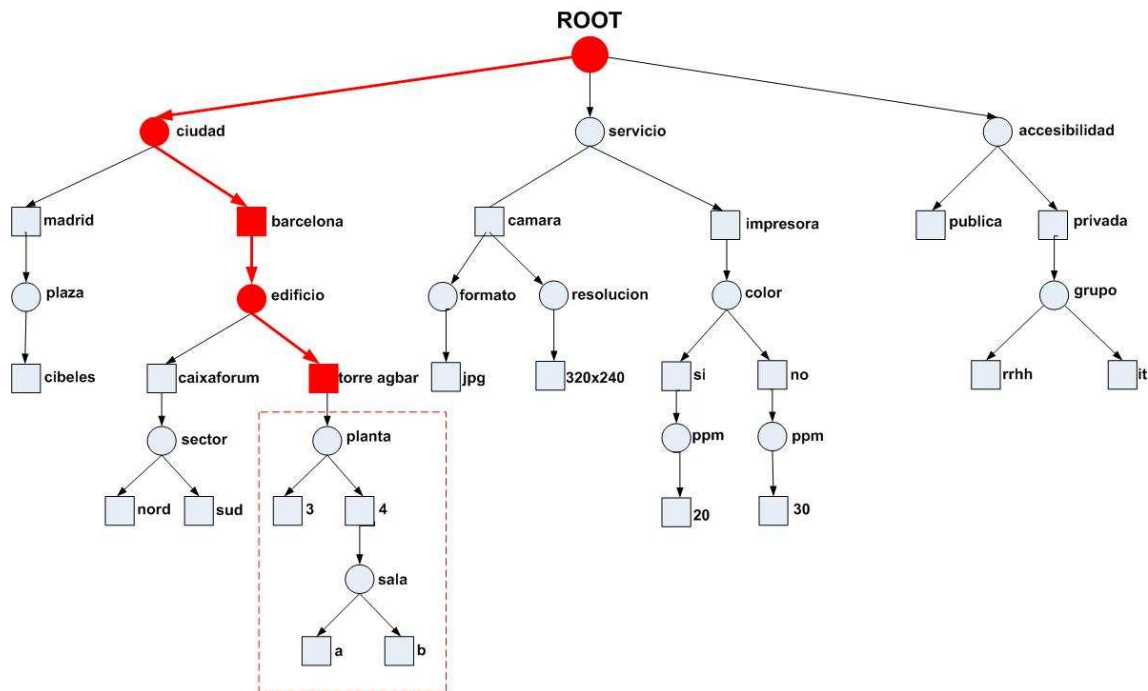


Fig. 1.16 *Búsqueda menos restrictiva*

El servidor volvería a hacer el recorrido descrito en el identificador enviado, pero el recorrido de este caso, solo llegaría hasta el valor “torre agbar”.

Como no es un valor final, sino que de allí salen más ramas, los resultados que se devolverían de esta búsqueda, serían todos los que hay en ese valor y en todos los otros valores que contienen las ramas que salen de ahí, es decir, la de cada una de las plantas y de cada una de las salas. Si un mismo resultado estuviera repetido, este solo se devolvería una vez.

En un último caso de que se quisiera hacer una búsqueda mucho más restrictiva, podría pedir que solo se le informará de los servicios que son una impresora a color, de acceso privado para IT, y que estuvieran situados en la sala b de la planta 4 de la torre Agbar de Barcelona:

```
[ciudad = barcelona [edificio = torre agbar [planta = 4 [sala = b]]]]
[servicio = impresora [color = si]]
[accesibilidad = privada [grupo = it]]
```

Fig. 1.17 *Identificador*

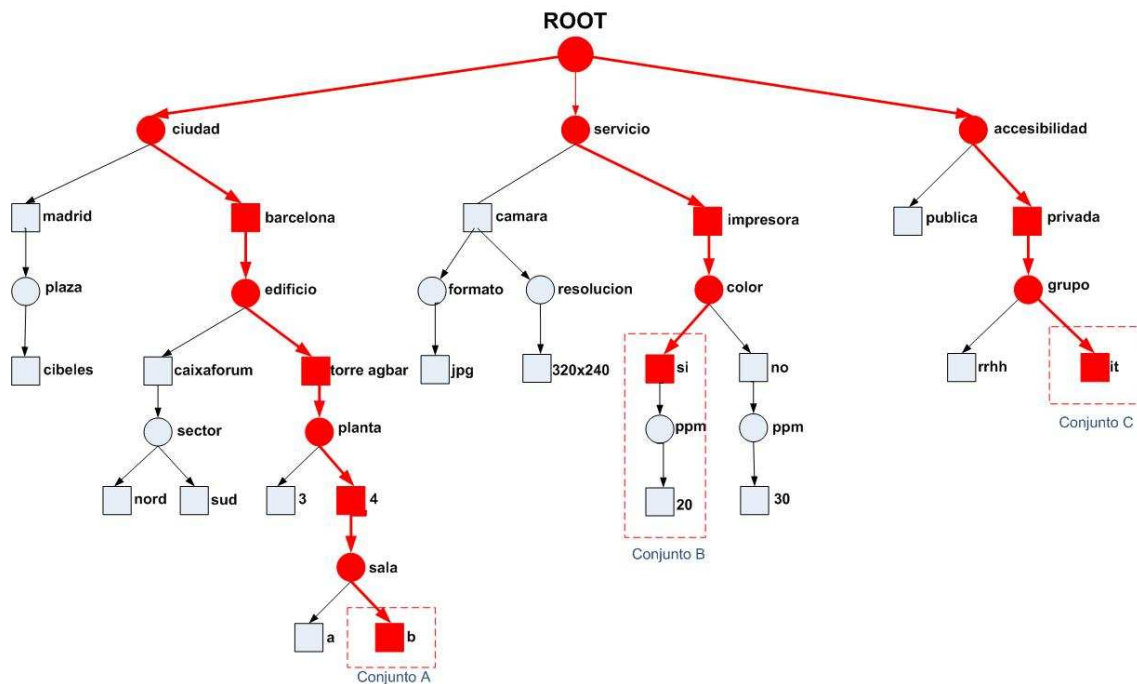


Fig. 1.18 *Búsqueda por varias ramas*

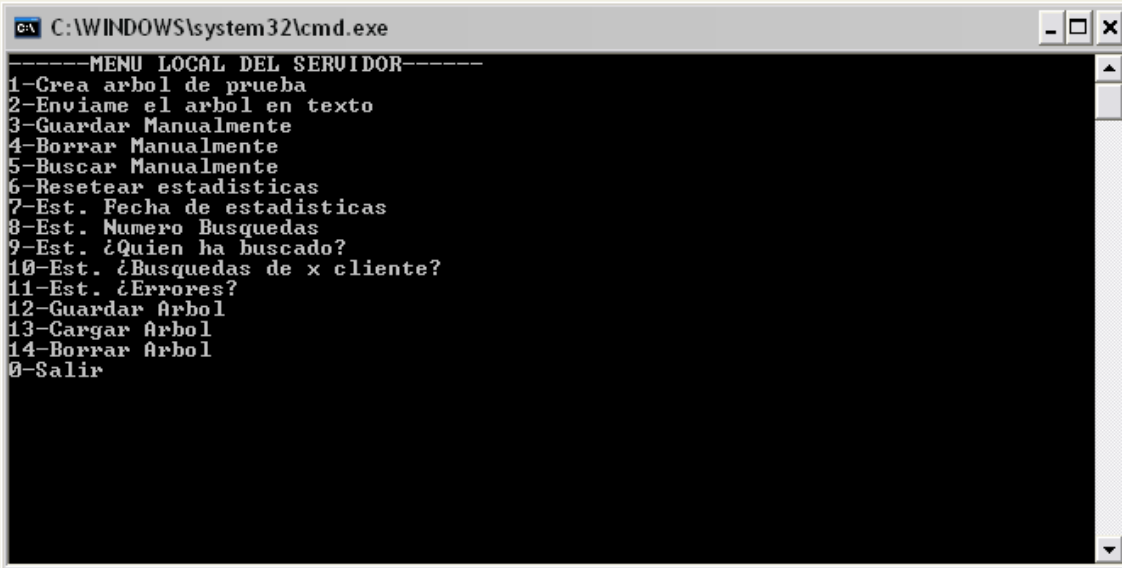
Como se observa en esta búsqueda, el servidor haría tres recorridos por el árbol, obteniendo 3 conjuntos distintos o no, de resultados posibles.

Una vez obtenidos estos 3 conjuntos, se procedería a comprobar cual o cuales de los resultados se repiten en los tres conjuntos, porque estos resultados serian los que se adecuan a la búsqueda pedida mediante el identificador, y estos serian los que se devolverían al cliente que ha hecho la búsqueda.

CAPÍTULO 2. IMPLEMENTACIÓN

2.1. Servidor

El servidor, es un programa, que en principio no tendría ningún entorno visual, y funcionaría en segundo plano, ya que se controlaría y usaría de forma remota. Aun así, se le implementa un menú en modo consola, para poder gestionarlo localmente para la realización de pruebas.



```
C:\WINDOWS\system32\cmd.exe

-----MENU LOCAL DEL SERVIDOR-----
1-Crea arbol de prueba
2-Envíame el arbol en texto
3-Guardar Manualmente
4-Borrar Manualmente
5-Buscar Manualmente
6-Resetear estadísticas
7-Est. Fecha de estadísticas
8-Est. Numero Búsquedas
9-Est. ¿Quién ha buscado?
10-Est. ¿Búsquedas de x cliente?
11-Est. ¿Errores?
12-Guardar Arbol
13-Cargar Arbol
14-Borrar Arbol
0-Salir
```

Fig. 2.1 Captura de la consola del servidor

La implementación del servidor, se divide en dos partes que están muy relacionadas. Por un lado hay las funciones centrales que hacen funcionar al servidor, y por otro lado las funciones que comunican el servidor con los distintos tipos de clientes que existen.

2.1.1. Funciones centrales

2.1.1.1 Implementación del árbol

La parte fundamental y más importante del servidor, es la generación del árbol de información que guardara todos los atributos y valores que le lleguen mediante nombres identificadores.

Por esto, para lograr crear un sistema de guardado de información en forma de árbol, que sobretodo no esté limitado ni predefinido previamente, y que además

tenga la virtud de permitir autogenerarse y autoampliarse a medida que va recibiendo nombres identificadores, se ha ideado un árbol con dos tipos distintas de clases.

La primera clase, es una clase “Tree-Attribute”, que tendrá el nombre de un atributo, y dentro contendrá una o varias clases del segundo tipo llamadas “Tree-Value”. Cada una de estas últimas, llevarán el nombre de un valor, y dentro contendrán una lista con la IP de nameinfos, y podrán tener clases “Tree-Attribute”. El root como tal, es un “Tree-Value” de nombre “root”.

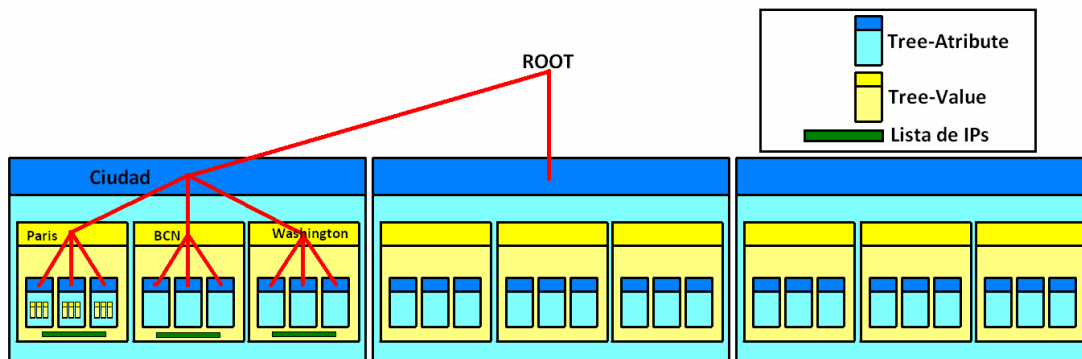


Fig. 2.2 Esquema de las clases que forman el árbol

Con estas clases se crea el árbol, y aparte, hay una tabla donde se guardan todos los nameinfos de los servicios que se han anunciado.

2.1.1.2 Prueba y reseteo del servidor

Para realizar pruebas en el árbol, se ha hecho una función que carga automáticamente en el árbol unos atributos y valores de prueba, junto con unos nameinfos.

Esta función permite que el administrador pueda probar el árbol sin tener que esperar que se anuncien servicios, o sin estar en red.

Si en algún momento se quiere resetear el servidor, hay una función para este caso, que borra todo el árbol y la lista de nameinfos, junto a todas las estadísticas de las que se hablará más adelante. De esta forma se deja el servidor limpio, sin tener que apagar y volver a abrirlo.

2.1.1.3 Guardar/Borrar nombres identificadores

Con la llegada de un paquete del servicio que se anuncia con su nombre identificador y nameinfo, este se pasa a la función de guardado.

Esta función añade el nameinfo a la lista, y transforma el nombre identificador a otro formato para seguidamente, proceder a crear si no estaban creadas antes, las ramas en el árbol con los atributos y valores que han llegado. En el último valor de cada rama que se crea, se añade a la lista de IPs, la que hace referencia al nameinfo que se ha guardado.

De la misma forma que llegan paquetes anunciando servicios, también llegan paquetes para borrar servicios. Cuando llega uno de estos, se pasa a la función de borrado, que elimina todas las referencias de ese servicio en el árbol, y el nameinfo al que corresponde.

2.1.1.4 Buscador

La finalidad del servidor, es poder encontrar lo que previamente se ha guardado, y aquí es donde entra esta función, el buscador.

Para tener un programar más completo y que sea mucho más útil, se han implementado dos tipos de buscador, el normal que hace la búsqueda a partir de un nombre identificador que se le envía, y el inverso que la hace desde una dirección IP.

El buscador normal, recibe un nombre identificador que ha generado la persona que quiere hacer la búsqueda, personalizándolo según sus intereses. Este, recorre todas las ramas que indique, y devuelve las IPs o nameinfos que se encuentren y que se ajusten a lo indicado.



Fig. 2.3 *Identificador de ejemplo y resultados que podría devolver*

Este buscador normal, tiene algunas particularidades hechas para ayudar en las búsquedas. Si queremos buscar todo lo que hay en Barcelona, pero en el árbol, por debajo continúan ramas con edificios, no hace falta escribir cada rama para buscar, sino que simplemente se pone `[ciudad = barcelona]`, y el buscador recorrerá todas las ramas que haya por debajo indicando los resultados.

Si da el caso que en la planta 4, salen ramas para salas y despachos, y solo queremos buscar lo que hay en las salas, en el nombre identificador, se busquen las salas con un `*` sin tener que nombrarlas todas.

```
[ciudad = barcelona [edificio = torre agbar [planta = 4[sala = * ]]]]
```

Fig. 2.4 Identificador

El buscador inverso, funciona al revés que el buscador normal. A este, se le pregunta por una IP, y recorre todas las ramas, para encontrar donde está guardada su referencia, y posteriormente transforma las ramas en las que se ha encontrado, en texto tipo [A=V [A=V]].



Fig. 2.5 Identificador

2.1.1.5 Guardar/Cargar árbol de desde archivo

En caso de cierre inesperado del servidor, y así evitar perder toda la información importante que hay guardada en el, se ha hecho una función que guarda todas las ramas del árbol, junto con las listas de IPs, en un archivo.



Fig. 2.6 Representación de guardar y cargar en árbol en un archivo

Con la función de cargar, se abre el archivo en el que hay la información guardada, y se va generando nuevamente el árbol, como si de mensajes de anuncio se tratase.

2.1.1.6 *Estadísticas*

Para llevar un control de lo que sucede en el servidor, y así conocer su uso, se ha implementado un sistema de estadísticas.

En cada búsqueda que se realice en el servidor, este la registrara juntamente con otras cosas que pasen como pueden ser errores.

Gracias a este sistema, el administrador que gestione el servidor, podrá ver la cantidad de búsquedas que se han hecho, y quien las ha hecho y también podrá consultar cuantas búsquedas ha hecho un cliente concreto.

Aparte de la información simplemente de uso, el sistema de estadísticas lleva un control de los errores que se han producido el servidor, indicando a qué hora se han producido, junto con la descripción del error.

Una vez se han revisado las estadísticas, el administrador tiene la opción de poder borrarlas o no.

2.1.1.7 *Otras*

Otra función implementada ha sido la de transformar todo el árbol a texto, para que el administrador, pueda pedir que se le muestre y así visualizar todas las ramas que tiene el árbol, junto con la lista de IPs que tiene cada rama. Gracias a esto, podrá testear mejor el servidor, para poder hacer las comprobaciones que cree necesarias.

Además se han creado funciones secundarias que son usadas por las funciones principales, como las que transforman los nombres identificadores a otros formatos más fáciles de utilizar por el servidor, o funciones que quitan caracteres innecesarios como por ejemplo pueden ser algunos espacios.

2.1.2. **Funciones de red**

Las funciones de red del servidor, son las que permiten recibir anuncios u ordenes desde el exterior, y una vez recibido el mensaje, lo pasan a la función central correspondiente. Las funciones de red implementadas, se agrupan en tres grupos, según al tipo de cliente con el que interactúan.

2.1.2.1 *Escucha de anuncios*

Como puede haber varios servidores, los mensajes que envían los servicios, son enviados periódicamente de manera UDP multicast a la IP "224.0.0.3", y hacia el puerto "1413". El servidor se subscribe a esa IP multicast, y va escuchando los mensajes, hasta encontrar los que son de anuncio.

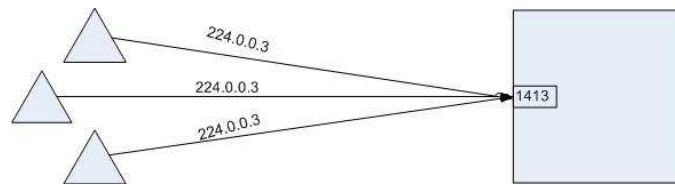


Fig. 2.7 *Socket de anuncio*

Los mensajes que se reciben del tipo anuncio desde los servicios, pueden ser para guardar uno nuevo, o borrarlo del servidor. Al recibir estos mensajes, se envían a las funciones de guardado o borrado, según corresponda.

2.1.2.2 Escucha de búsquedas

Los mensajes de pregunta y respuesta de búsqueda, son TCP. Las preguntas se reciben por el puerto "1414". Al recibirlos, esta capa mira qué tipo de búsqueda es, si normal o inversa, y se envía a la función de búsqueda correspondiente. Una vez obtenida la respuesta, se vuelve a enviar al cliente por TCP, hacia el puerto "1415".

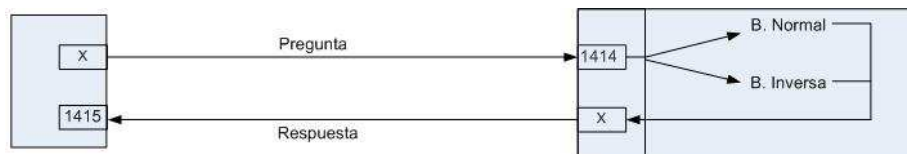


Fig. 2.8 *Sockets de búsqueda*

2.1.2.3 Escucha de gestión

El servidor se puede gestionar remotamente, desde un cliente de gestión que se explicara más adelante. Los mensajes, son TCP y se reciben por el puerto "1416". Una vez recibido, se analizan y se envían a la función correspondiente, para enviarle después la respuesta al cliente hacia el puerto "1417".

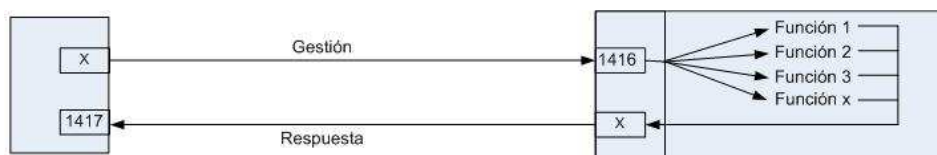


Fig. 2.9 *Sockets de gestión*

2.2. Clientes

Para utilizar el sistema implementado, se han realizado tres tipos de clientes. Un primer tipo es el cliente de anuncios, que emularía un servicio que se anuncia. Un segundo tipo es el cliente de búsqueda, que sirve para realizar búsquedas en el servidor. Por último, hay el cliente de gestión, que puede administrar el servidor.

2.2.1. Cliente de anuncio

El cliente de anuncio iría implementado dentro del servicio que se quisiera anunciar, ya sea una impresora, una cámara de video, o un sensor, pero para poder probarlo, se ha implementado uno de forma virtual.

Este, tiene guardada y muestra la ip multicast y el puerto hacia el que enviará los mensajes de anuncio. También coge y muestra la ip de la maquina en la que se ha inicializado, la cual será con la que se anunciará.

En la línea que hay para escribir, se pondrá el nombre identificador con el que queremos que se guarde en el árbol, y que será con lo que podrán hacer búsquedas para encontrara el servicio.

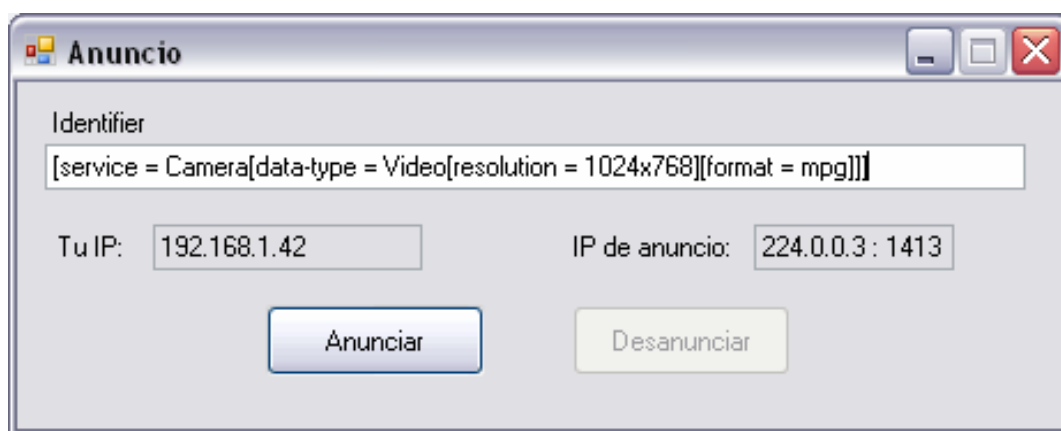


Fig. 2.10 Captura de la aplicación de anuncio

Una vez rellenado, se procederá a anunciar el servicio con el botón anunciar, y posteriormente se podrá seguir anunciando, o se podrá enviar un mensaje de desanuncio, para borrar ese servicio del servidor.

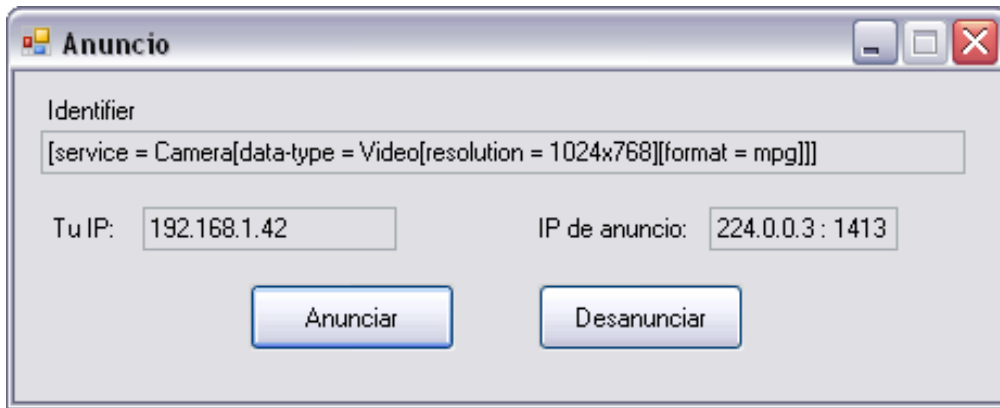


Fig. 2.11 Captura de la aplicación de anuncio

2.2.2. Cliente de búsqueda

Los clientes de búsqueda, permiten realizar peticiones de búsqueda remotamente. El cliente de búsqueda implementado, permite hacer las peticiones tanto desde un nombre identificador, como desde una IP.

Lo primero que se le tiene que indicar al cliente, es la ip del servidor en el que se quiere realizar la búsqueda. Una vez introducida, la persona que quiera buscar algo mediante un identificador, lo escribirá siguiendo la estructura de `[A=V[A=V]]`, y con las ayudas descritas anteriormente como utilizar `"*"` para buscar en todos los valores de un atributo.

Seguidamente, al presionar el botón, se enviará la petición al servidor y se esperará la respuesta con la o las IPs de los servicios que se corresponden y concuerdan con el identificador enviado.

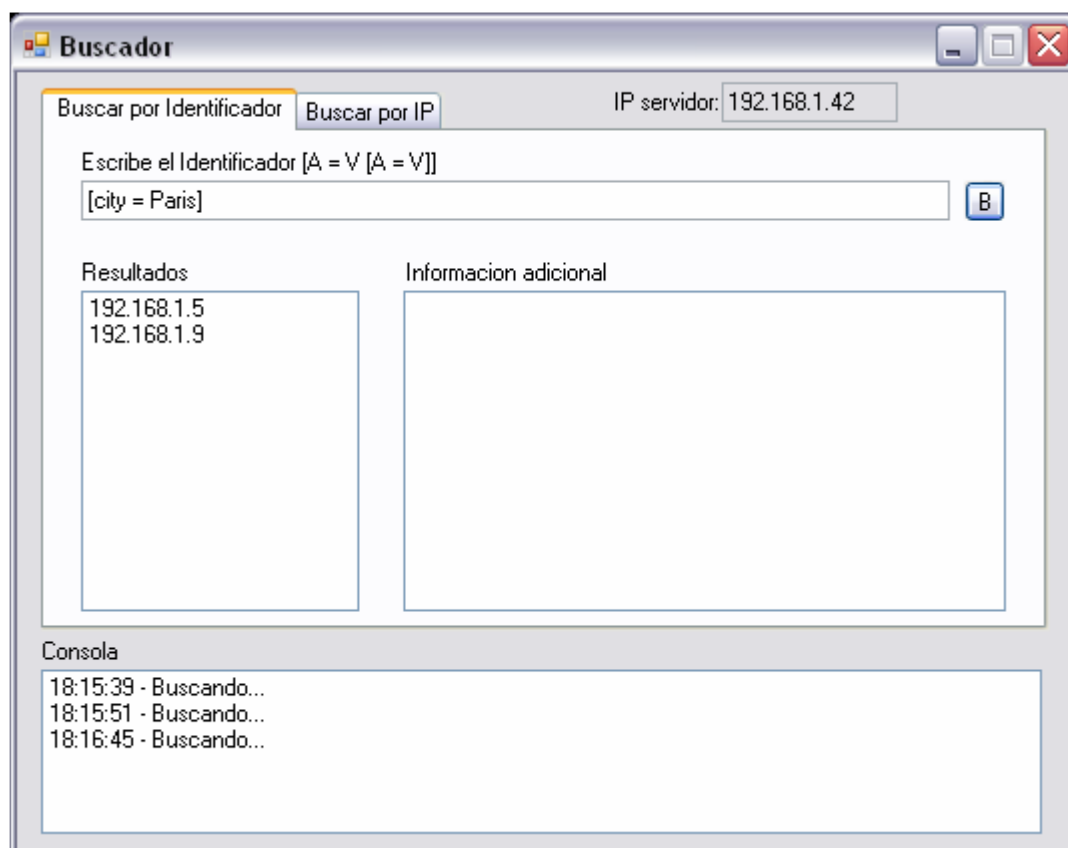


Fig. 2.12 Captura de la aplicación de búsquedas

En la imagen anterior, se observa una pequeña búsqueda que se ha realizado con un identificador “[city = Paris]”, que ha devuelto dos IPs como resultado con valores “192.168.1.5” y “192.168.1.9”.

Si algún resultado tiene información extra, esta se mostrara en la zona de información adicional, y en un “log” en la parte inferior del programa, se irá mostrando lo que va ocurriendo en el cliente, o si no se han encontrado respuestas.

Seleccionando la otra pestaña en el cliente, se va a la opción de búsquedas mediante una IP, es decir, búsquedas inversas.

En este caso, se introduce una IP de un servicio del que queremos información, y se hace la petición, cuya respuesta será todas las ramas en las que se encuentra referenciada la IP de ese servicio.

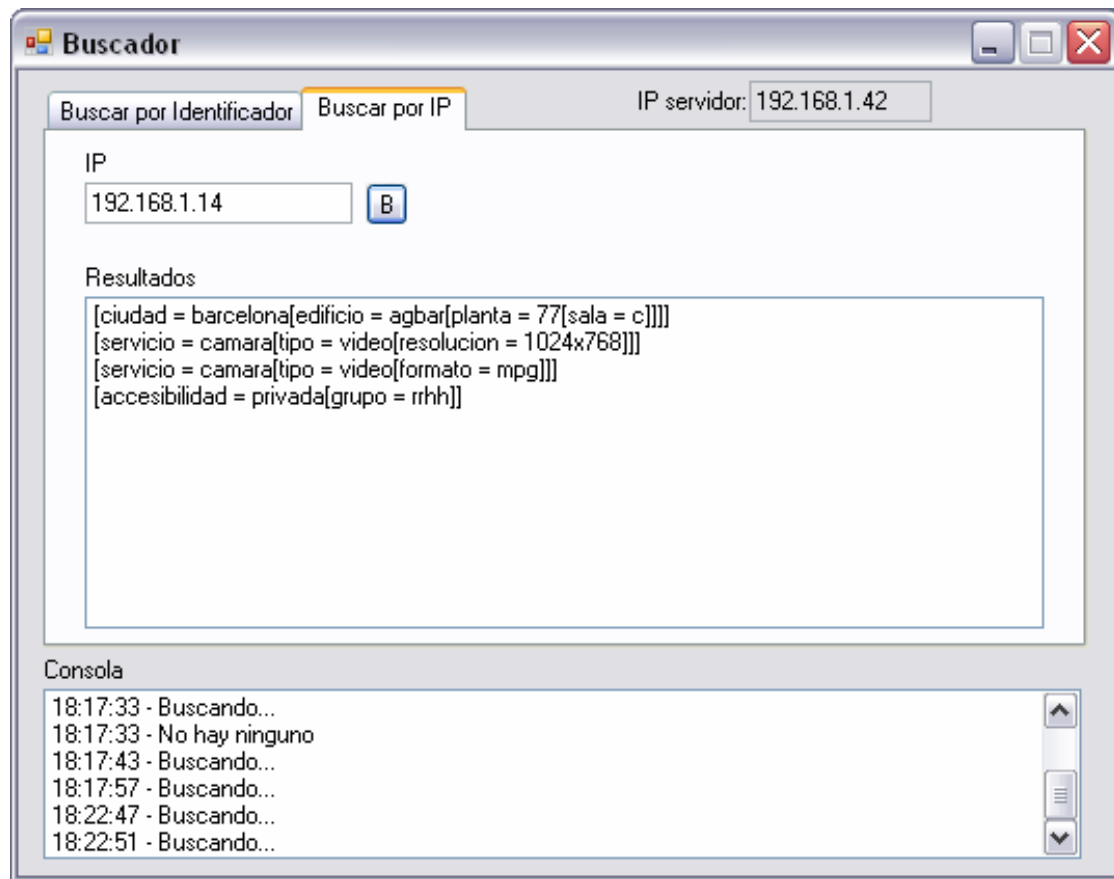


Fig. 2.13 Captura de la aplicación de búsquedas

En el ejemplo de la imagen anterior, se pregunta por la IP “192.168.1.14”, y el servidor nos devuelve cuatro resultados, que son las cuatro ramas donde se encuentra referenciada esa IP.

2.2.3. Cliente de gestión

El cliente de gestión, sería utilizado por los administradores del servidor, y permite obtener mucha información y configurar el servidor también, todo remotamente desde otro ordenador.

En este cliente, lleva incluido los otros dos clientes mencionados anteriormente, el que permite guardar y borrar servicios, y el que permite hacer las distintas búsquedas.

Además de llevar incluidos estos dos clientes, también permite conocer las estadísticas guardadas en el servidor y actuar en el árbol de información igual que se puede hacer desde el menú local.

2.2.3.1 Pestaña árbol

Una vez hemos introducido la Ip del servidor a gestionar, encontramos una primera pestaña en la que podemos ver o modificar el árbol.

En esta primera pestaña, tenemos la opción de ver todas las ramas que hay en el árbol, junto con las IPs que tienen referenciadas.

Si se quiere resetear toda la información del servidor, se hará con el botón “Borra árbol”, y para cargar información otra vez al servidor, se puede hacer creando un árbol de prueba con atributos y valores predefinidos, o cargando la información desde un archivo con el botón “Carga árbol”.

Este archivo, se habrá creado anteriormente con el botón “Guardar árbol”, que transforma la información del servidor en un archivo de texto.

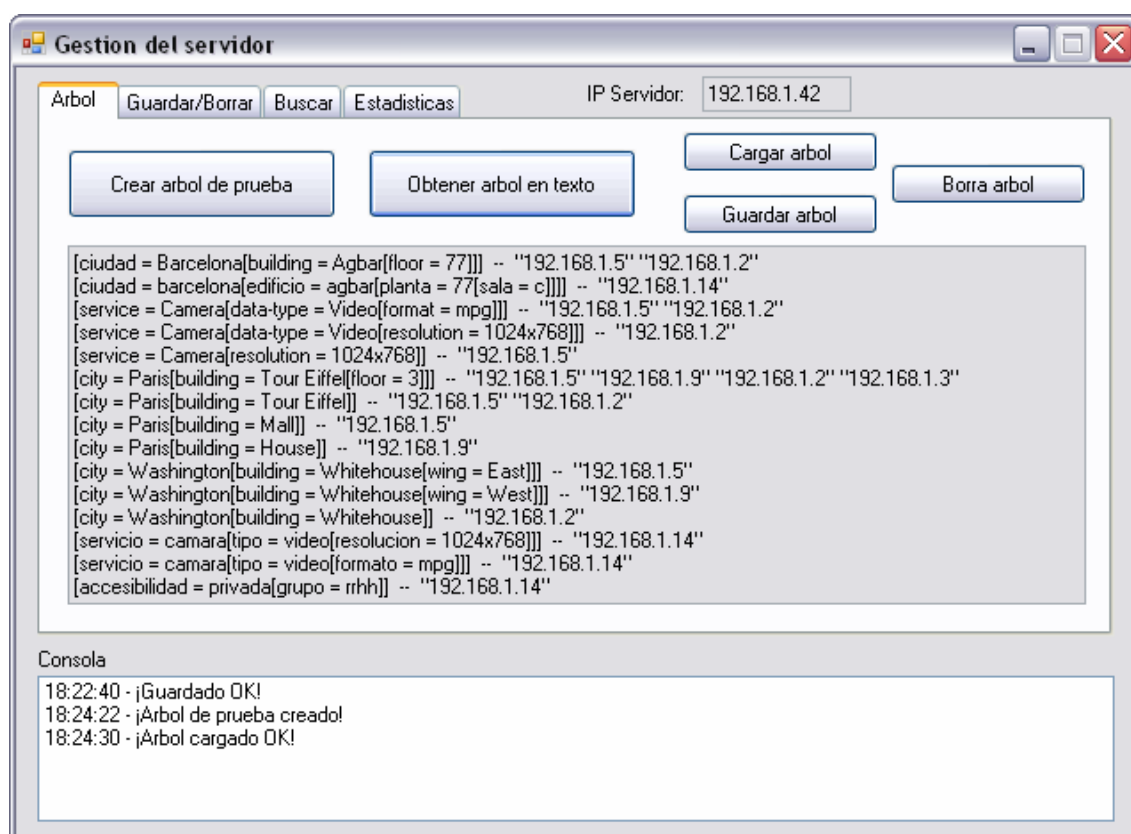


Fig. 2.14 Captura de la aplicación de gestión en la pestaña “árbol”

En la parte inferior del programa, igual que en el cliente buscador, se ha implementado un log, que irá informando de todo lo que vaya haciendo el cliente de gestión, indicando también la hora exacta en la que se ha producido cada evento.

2.2.3.2 Pestaña Guardar/Borrar

La siguiente pestaña, nos la tenemos que imaginar como si fuera un cliente de anuncio, ya que nos permite hacer las mismas cosas.

Por un lado, escribimos un nombre identificador, y la IP del servicio que queremos guardar, y enviamos un mensaje de anuncio, pero esta vez TCP, y solo para el servidor que estamos gestionando.

Indicando una IP, también podemos borrar ese servicio del servidor en el que estamos conectados.

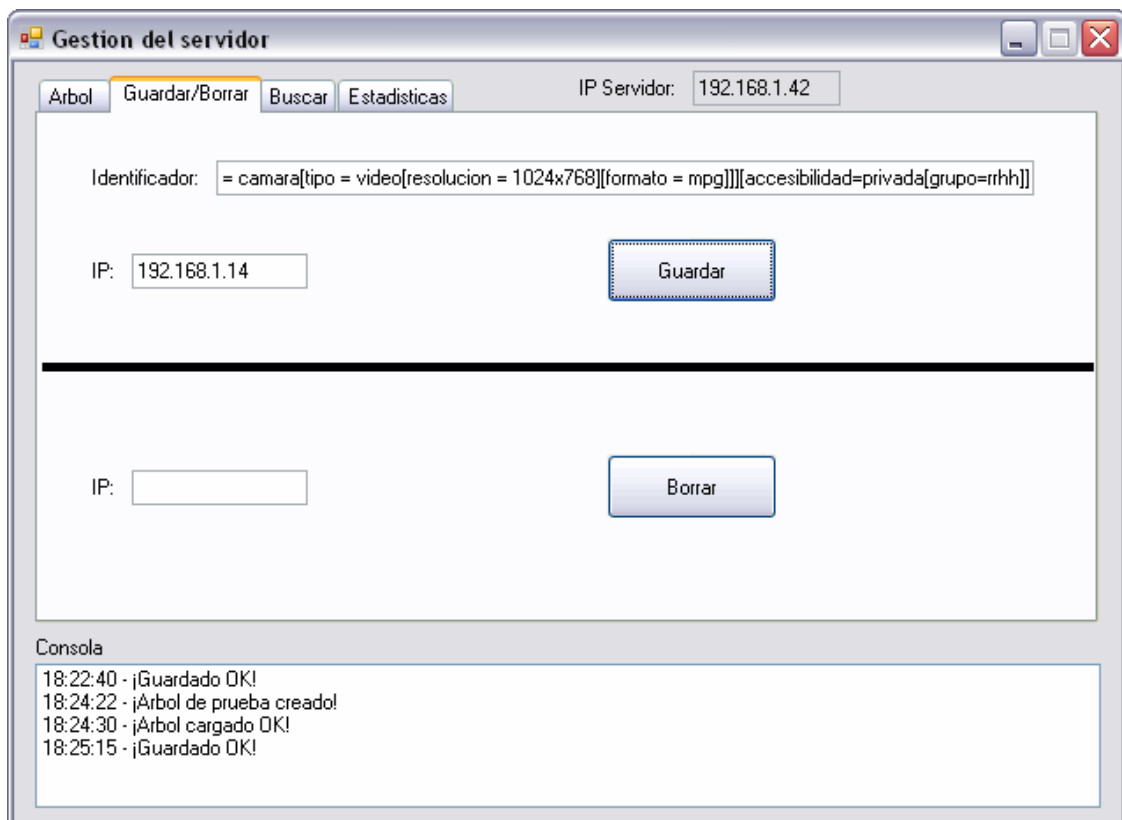


Fig. 2.15 Captura de la aplicación de gestión en la pestaña “guardar/borrar”

2.2.3.3 Pestaña buscar

La tercera pestaña, es un cliente de búsqueda, permitiéndonos buscar de forma normal o inversa.

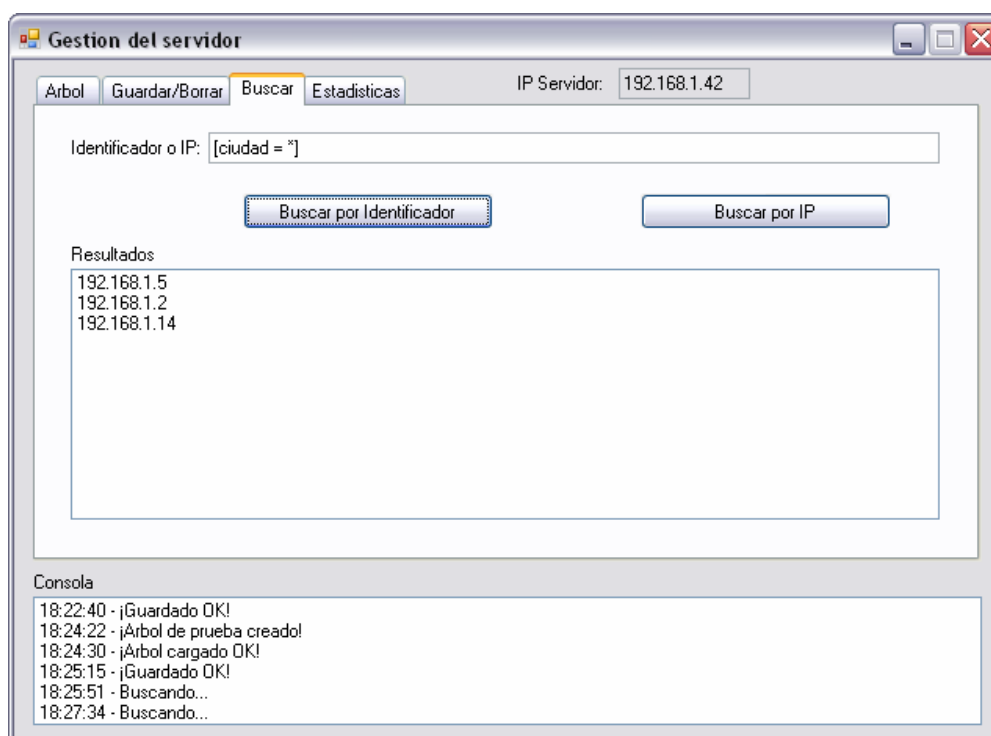


Fig. 2.16 Captura de la aplicación de gestión en la pestaña “buscar”

Para realizar estas búsquedas, tan solo tenemos que introducir en identificador adecuado a nuestros intereses, o la IP, y presionar el botón correspondiente a nuestro tipo de búsqueda, para conocer automáticamente los resultados.

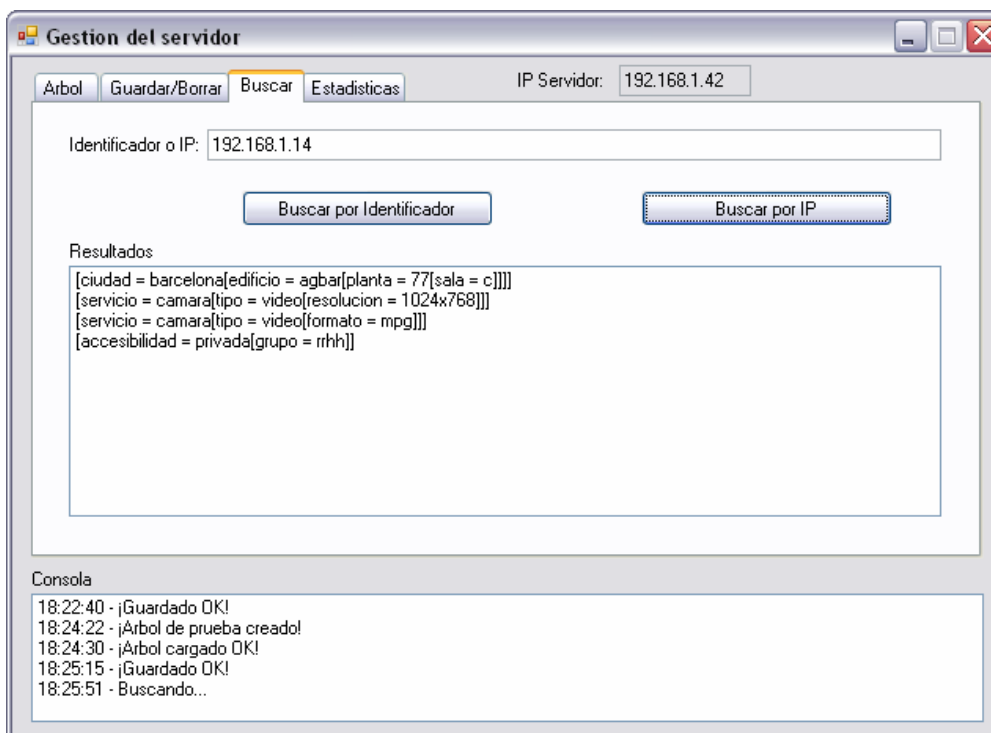


Fig. 2.17 Captura de la aplicación de gestión en la pestaña “buscar”

2.2.3.4 Pestaña estadísticas

La ultima pestaña de que dispone el cliente de gestión, es la que nos muestra las estadísticas.

Esta pestaña es muy útil ya que podemos conocer si se usa o no el servidor teniendo en cuenta el numero de búsquedas realizadas, y también permite conocer quien ha realizado las búsquedas y cuantas búsquedas ha hecho ese cliente.

Si hay muchos clientes que han buscado, para buscar más rápidamente y conocer el número de búsquedas que ha hecho un cliente en concreto, disponemos de la opción para buscar un solo cliente.

Además, para que el administrador pueda saber si el servidor funciona correctamente, las estadísticas también nos muestran una ventana con los distintos errores que ha habido en el servidor, con la hora en que se ha producido cada uno de los errores, y la descripción del mismo.

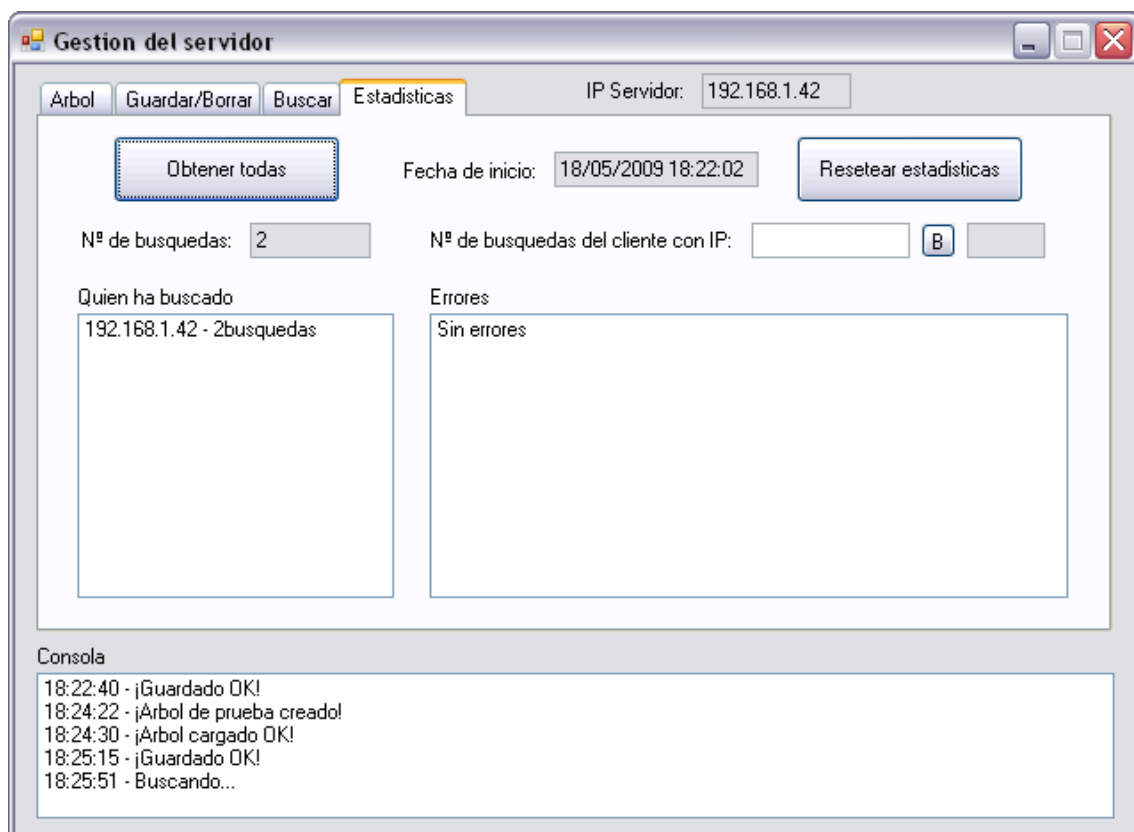


Fig. 2.18 Captura de la aplicación de gestión en la pestaña "estadísticas"

CAPÍTULO 3. CASO PRÁCTICO

3.1. Empresa textil

Después de describir e implementar ese sistema intencional, para poder entender mejor cómo se podría aplicar este sistema y los usos que podría tener en el mundo real, se explicará mediante un ejemplo un posible uso.

Supongamos una empresa textil, que tiene conectadas mediante una NetLAN, unas oficinas centrales en la ciudad de Barcelona, otras oficinas en Madrid, y además varias tiendas repartidas por España, Portugal, Italia y Francia del siguiente modo:

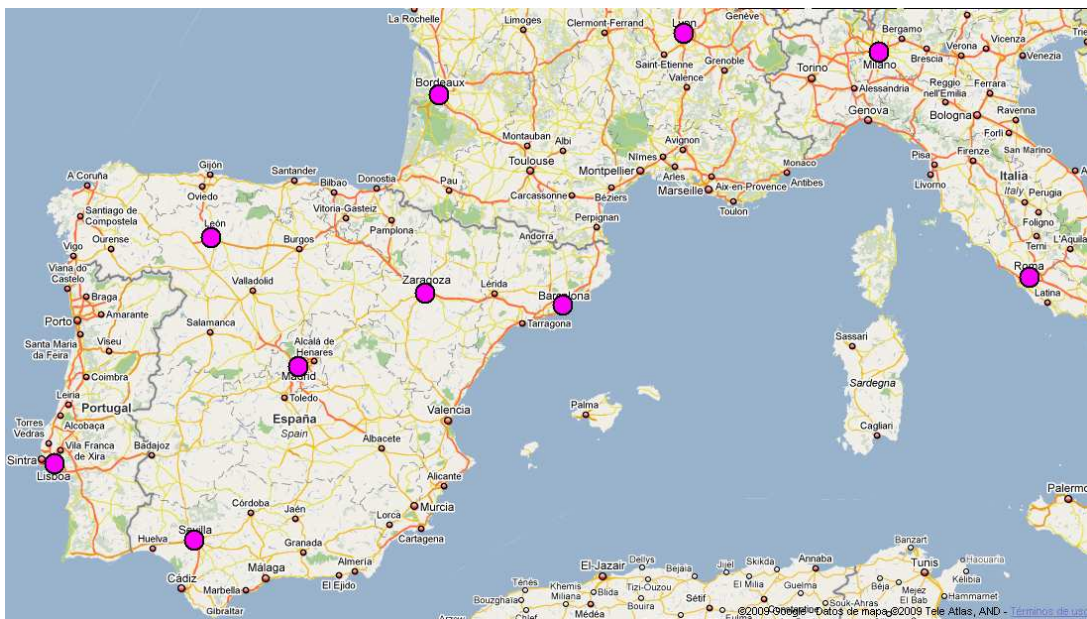


Fig. 3.1 Localización de las tiendas

El número de tiendas y oficinas que tendría esta empresa sería:

- 1 oficina central en Barcelona
- 1 oficina en Madrid
- 3 tiendas en Barcelona
- 2 tiendas en Madrid
- 1 tienda en Zaragoza
- 1 tienda en León
- 1 tienda en Sevilla
- 1 tienda en Lisboa
- 1 tienda en Bordeaux
- 1 tienda en Lyon
- 1 tienda en Milán
- 2 tiendas en Roma

Las oficinas de Barcelona y Madrid, tienen varias impresoras, cámaras de seguridad, y sensores de temperatura ubicados de la siguiente forma:

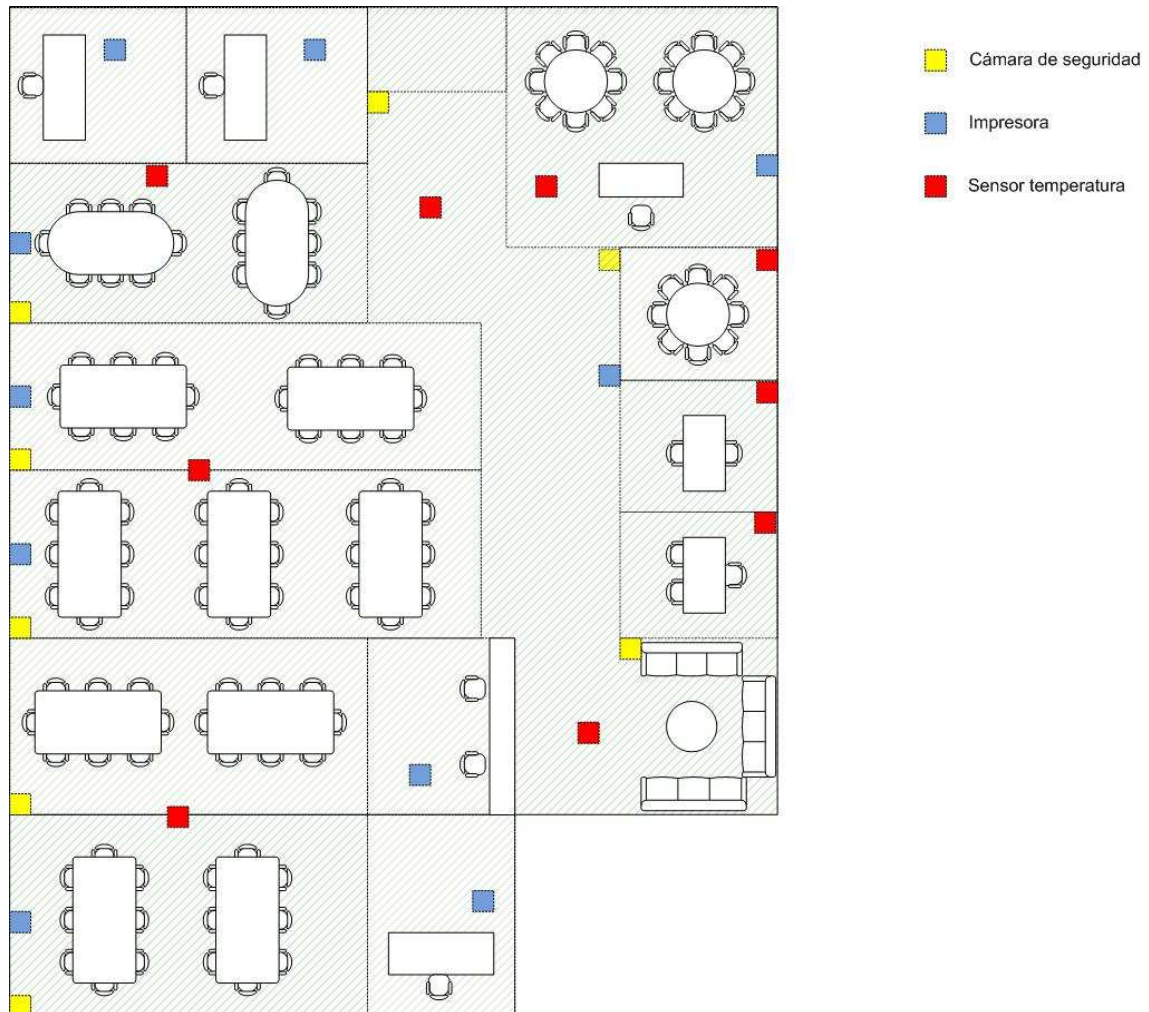


Fig. 3.2 Plano de las oficinas centrales de Barcelona

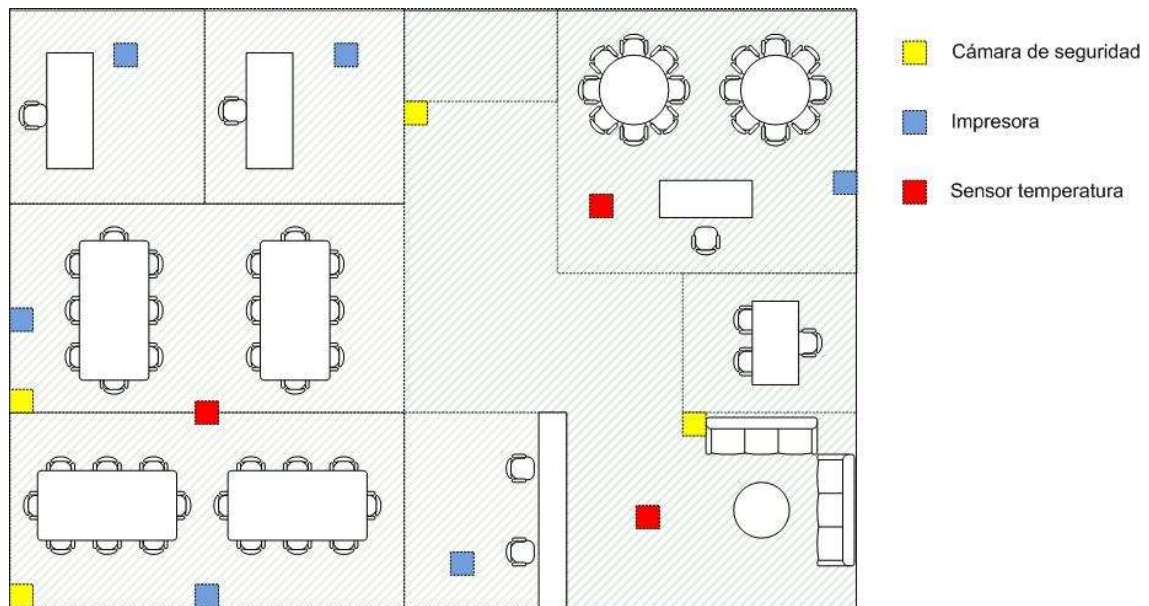


Fig. 3.3 *Plano de las oficinas de Madrid*

En el caso de las tiendas, todas tienen la misma forma, con un despacho, almacén y zona de tienda, y dentro tienen impresoras, cámaras de vigilancia, sensores de temperatura y contadores de la gente que entra en la tienda:

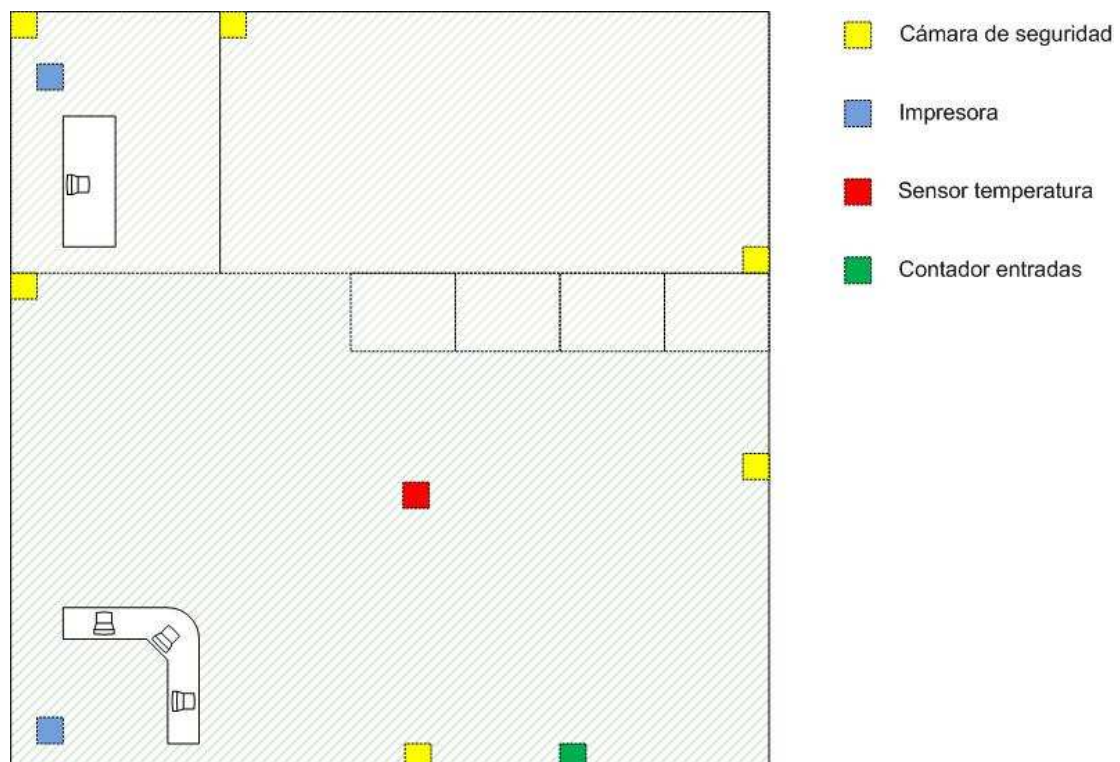


Fig. 3.4 *Plano de las tiendas*

El recuento final de servicios a anunciar sería:

- Total en todas las oficinas:
 - 12 cámaras de seguridad
 - 16 impresoras
 - 12 sensores de temperatura
- Total en todas las tiendas:
 - 84 cámaras de seguridad
 - 28 impresoras
 - 14 sensores de temperatura
 - 14 contador entradas

Para crear los nombres identificadores de los distintos servicios y anunciarlos por la red, seguiremos un esquema en el que se describirá la ubicación, el tipo de servicio y algunas de sus características principales, y la accesibilidad que se tienen a este servicio:

[país = X [ciudad = X [calle = X]]]
[espacio = X [zona = X [otro = X]]]
[servicio = X [otro = X [otro = X]]]
[accesibilidad = X [grupo = X]]

Fig. 3.5 *Formato del identificador a usar*

Siguiendo el esquema anterior, la parte del nombre que describe la ubicación física sería igual en todos los dispositivos de una misma tienda u oficina, y para cada tienda/oficina sería:

- Oficina central en Barcelona
 - *[país = España [ciudad = Barcelona [calle = Diputacio]]]*
- Oficina en Madrid
 - *[país = España [ciudad = Madrid [calle = Serrano]]]*
- Tienda 1 en Barcelona
 - *[país = España [ciudad = Barcelona [calle = Diagonal]]]*
- Tienda 2 en Barcelona
 - *[país = España [ciudad = Barcelona [calle = Ramblas]]]*
- Tienda 3 en Barcelona
 - *[país = España [ciudad = Barcelona [calle = Aribau]]]*
- Tienda 1 en Madrid
 - *[país = España [ciudad = Madrid [calle = Ayala]]]*

- Tienda 2 en Madrid
 - *[país = España [ciudad = Madrid [calle = Alcala]]]*
- Tienda en Zaragoza
 - *[país = España [ciudad = Zaragoza [calle = Coso]]]*
- Tienda en León
 - *[país = España [ciudad = León [calle = Roma]]]*
- Tienda en Sevilla
 - *[país = España [ciudad = Sevilla [calle = Oriente]]]*
- Tienda en Lisboa
 - *[país = Portugal [ciudad = Lisboa [calle = Dos Sapateiros]]]*
- Tienda en Bordeaux
 - *[país = Francia [ciudad = Bordeaux [calle = Du Loup]]]*
- Tienda en Lyon
 - *[país = Francia [ciudad = Lyon [calle = Dubois]]]*
- Tienda en Milano
 - *[país = Italia [ciudad = Milano [calle = Buonaparte]]]*
- Tienda 1 en Roma
 - *[país = Italia [ciudad = Roma [calle = Lambro]]]*
- Tienda 2 en Roma
 - *[país = Italia [ciudad = Roma [calle = Simeto]]]*

La parte del nombre que hace referencia al tipo de espacio, es decir si es tienda u oficina, y que lugar dentro de estos, sería igual para cada servicio dentro del mismo tipo de espacio, y los distintos posible que se podrían dar serian los siguientes:

- Oficinas
 - *[espacio = oficina [departamento = IT [zona = general]]]*
 - *[espacio = oficina [departamento = IT [zona = despacho1]]]*
 - *[espacio = oficina [departamento = RRHH [zona = general]]]*
 - *[espacio = oficina [departamento = Administración [zona = general]]]*
 - *[espacio = oficina [departamento = Diseño [zona = general]]]*
 - *[espacio = oficina [departamento = Gerencia [zona = general]]]*
 - *[espacio = oficina [departamento = Gerencia [zona = despacho1]]]*
 - *[espacio = oficina [departamento = Gerencia [zona = despacho2]]]*
 - *[espacio = oficina [departamento = Varios [zona = entrada]]]*
 - *[espacio = oficina [departamento = Varios [zona = pasillo]]]*
 - *[espacio = oficina [departamento = Varios [zona = sala1]]]*
 - *[espacio = oficina [departamento = Varios [zona = sala2]]]*
 - *[espacio = oficina [departamento = Varios [zona = sala3]]]*
 - *[espacio = oficina [departamento = Varios [zona = sala4]]]*

- Tiendas
 - *[espacio = tienda [zona = almacén]]*
 - *[espacio = tienda [zona = despacho]]*
 - *[espacio = tienda [zona = ventas]]*

Para crear el nombre identificador, lo siguiente sería la descripción del tipo de servicio. En este ejemplo que se realiza, se supone que todas las cámaras de vigilancia son iguales, que todas las impresoras son iguales, y lo mismo con los sensores de temperatura y contadores de entrada:

- Cámaras de vigilancia:
 - *[servicio = cámara [tipo de datos = video [formato = mpg]]
[resolución = 1024x768]]*
- Impresoras:
 - *[servicio = impresora [color = si]
[ppm = 10]
[ppp = 600]]*
- Sensores de temperatura:
 - *[servicio = sensor [tipo = temperatura]]*
- Contadores de entrada:
 - *[servicio = contador [medida = clientes]]*

Por último, la parte que finaliza el nombre, sería la de accesibilidad, y referente al ejemplo que estamos haciendo, las posibles serían:

- Accesibilidad pública:
 - *[accesibilidad = publica]*
- Accesibilidad privada:
 - *[accesibilidad = privada [departamento = Tiendas]]*
 - *[accesibilidad = privada [departamento = IT]]*
 - *[accesibilidad = privada [departamento = RRHH]]*
 - *[accesibilidad = privada [departamento = Administración]]*
 - *[accesibilidad = privada [departamento = Diseño]]*
 - *[accesibilidad = privada [departamento = Gerencia]]*

Los nombres identificadores completos que utilizarían los servicios para anunciarse por la red local de la empresa, se compondrían de las posibles partes que se acaban de describir. Para verlo más claro, a continuación se describen los nombres identificadores de los servicios de una tienda concreta:

- Impresora del despacho de la tienda de Sevilla:
 - *[país = España [ciudad = Sevilla [calle = Oriente]]]
[espacio = tienda [zona = despacho]]
[servicio = impresora [color = si]
[ppm = 10]
[ppp = 600]]
[accesibilidad = privada [departamento = Tiendas]
[departamento = IT]]*
- Impresora de la zona de ventas de la tienda de Sevilla:
 - *[país = España [ciudad = Sevilla [calle = Oriente]]]
[espacio = tienda [zona = ventas]]
[servicio = impresora [color = si]
[ppm = 10]
[ppp = 600]]
[accesibilidad = privada [departamento = Tiendas]
[departamento = IT]]*
- Sensor de temperatura de la tienda de Sevilla:
 - *[país = España [ciudad = Sevilla [calle = Oriente]]]
[espacio = tienda [zona = ventas]]
[servicio = sensor [tipo = temperatura]]
[accesibilidad = privada [departamento = Tiendas]
[departamento = IT]]*
- Contador de entradas de la tienda de Sevilla:
 - *[país = España [ciudad = Sevilla [calle = Oriente]]]
[espacio = tienda [zona = ventas]]
[servicio = contador [medida = clientes]]
[accesibilidad = privada [departamento = Tiendas]
[departamento = IT]
[departamento = Gerencia]]*
- Cámaras de la zona de ventas de la tienda de Sevilla:
 - *[país = España [ciudad = Sevilla [calle = Oriente]]]
[espacio = tienda [zona = ventas]]
[servicio = cámara [tipo de datos = video [formato = mpg]]
[resolución = 1024x768]]
[accesibilidad = privada [departamento = Tiendas]
[departamento = IT]]*

El árbol generado por el anuncio de la impresora de la zona de ventas de la tienda de Sevilla junto con su nameinfo, tendría el siguiente aspecto:

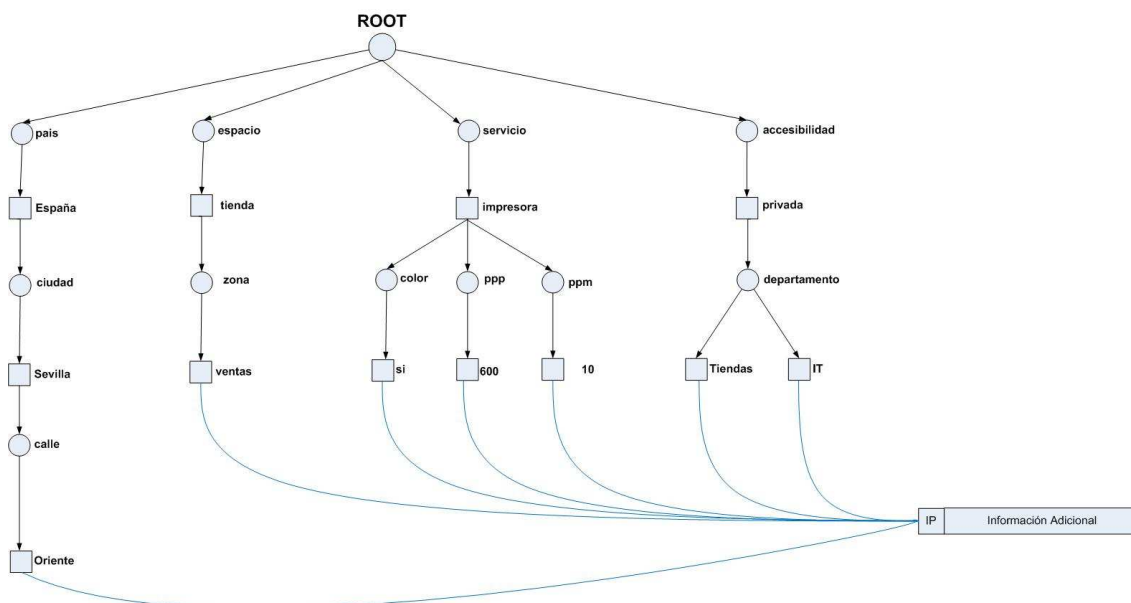


Fig. 3.7 Árbol generado por el anuncio de la impresora

Y si una vez generado el árbol anterior, llegan todos los anuncios de todas las cámaras de seguridad de la tienda de Sevilla, el árbol quedaría actualizado de la siguiente forma:

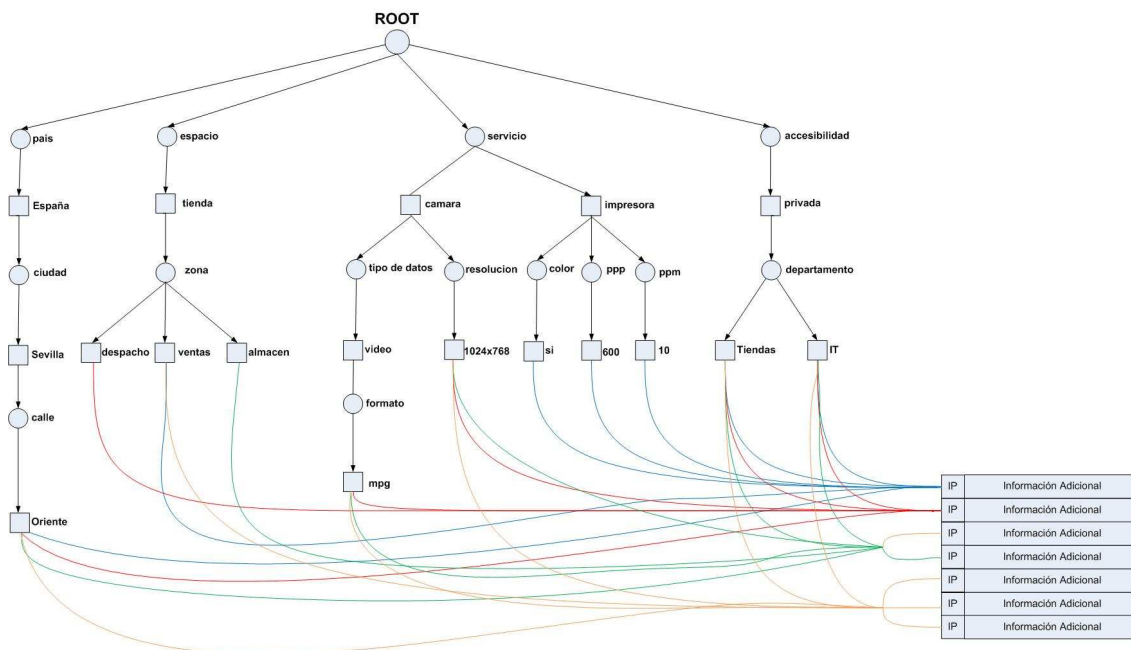


Fig. 3.8 Árbol con el anuncio de la impresora y de las cámaras de seguridad

Finalmente, con la llegada de todos los anuncios de todos los servicios de la red de tiendas y oficinas, se generaría un árbol de información muy grande y muy útil de atributos y valores, cuya forma sería:

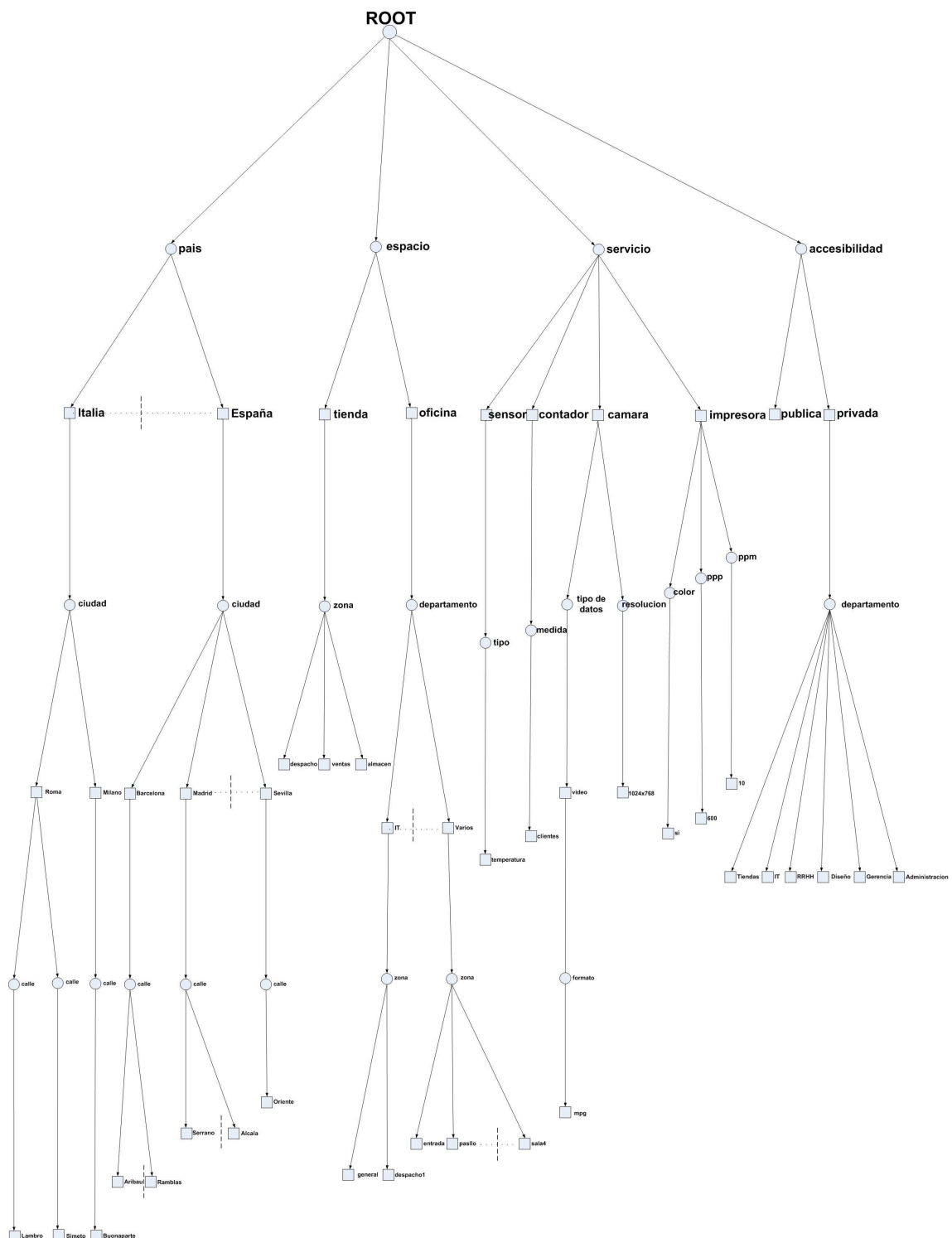


Fig. 3.9 Árbol formado por todos los identificadores de los servicios

Cuando algún cliente quisiera buscar algunos servicios, crearía un nombre identificador para la búsqueda constituido por los atributos y valores de las características que le interesase. Veamos algunos ejemplos:

Búsqueda de todo lo que hay en Barcelona:

- *[país = España [ciudad = Barcelona]]*

Búsqueda de todo lo que hay en la calle Aribau de Barcelona

- *[país = España [ciudad = Barcelona [calle = Aribau]]]*

Búsqueda de las impresoras a color y de 10 ppm

- *[servicio = impresora [color = si]
[ppm = 10]*

Búsqueda de las impresoras de las tiendas:

- *[servicio = impresora]
[espacio = tienda]*

Búsqueda de las cámaras de video de las tiendas de Barcelona:

- *[país = España [ciudad = Barcelona]]
[espacio = tienda]
[servicio = cámara [tipo de datos = video]]*

Búsqueda de las impresoras a color de Madrid con acceso privado para IT

- *[país = España [ciudad = Madrid]]
[servicio = impresora [color = si]]
[accesibilidad = privada [departamento = IT]]*

Búsqueda de las impresoras a color y 600ppp del pasillo de las oficinas ubicadas en Barcelona a las que tengan acceso privado los departamentos de RRHH y IT:

- *[país = España [ciudad = Barcelona]]
[servicio = impresora [color = si]
[ppp = 600]]
[accesibilidad = privada [departamento = IT]
[departamento = RRHH]]*

CAPÍTULO 4. PRUEBAS

4.1. Uso de red

Con el programa Wireshark, analizaremos los paquetes que se envían por la red, referentes a nuestro sistema. Por un lado mediremos los paquetes multicast de anuncio de los servicios, y por otro los paquetes de búsqueda, teniendo en cuenta tanto los de pregunta como los de respuesta. Para el caso de las respuestas, consideraremos que el servidor nos devuelve solamente un resultado.

Se tiene que tener en cuenta que el tamaño de estos paquetes varía según la extensión del nombre identificador tanto de anuncio como de búsqueda, y lo mismo pasa con los mensajes de respuesta. Por esta causa, las medidas las realizaremos enviando nombres identificadores con una extensión media, con el siguiente texto:

```
[país = España [ciudad = Barcelona]]
[servicio = impresora [color = si]
[ppp = 600]]
[accesibilidad = privada [departamento = IT]
[departamento = RRHH]]
```

En analizador de paquetes Wireshark, captura los paquetes y nos los muestra describiendo cada cabecera y contenido, para así obtener la medida.

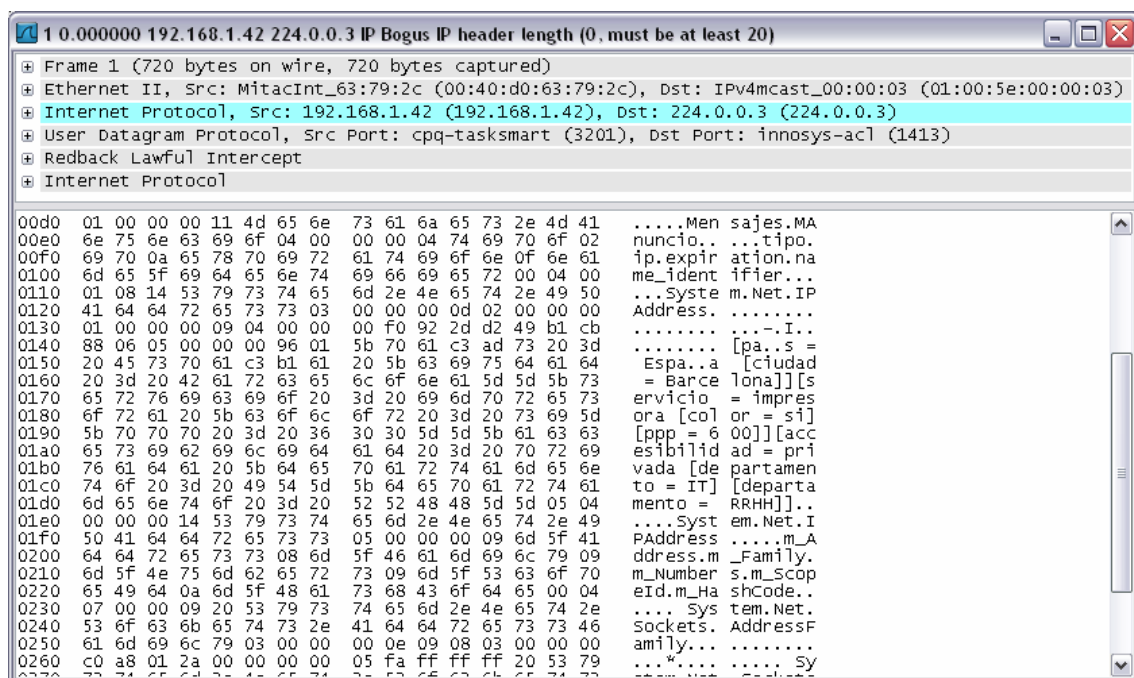


Fig. 4.1 Captura de un paquete de anuncio con el programa Wireshark

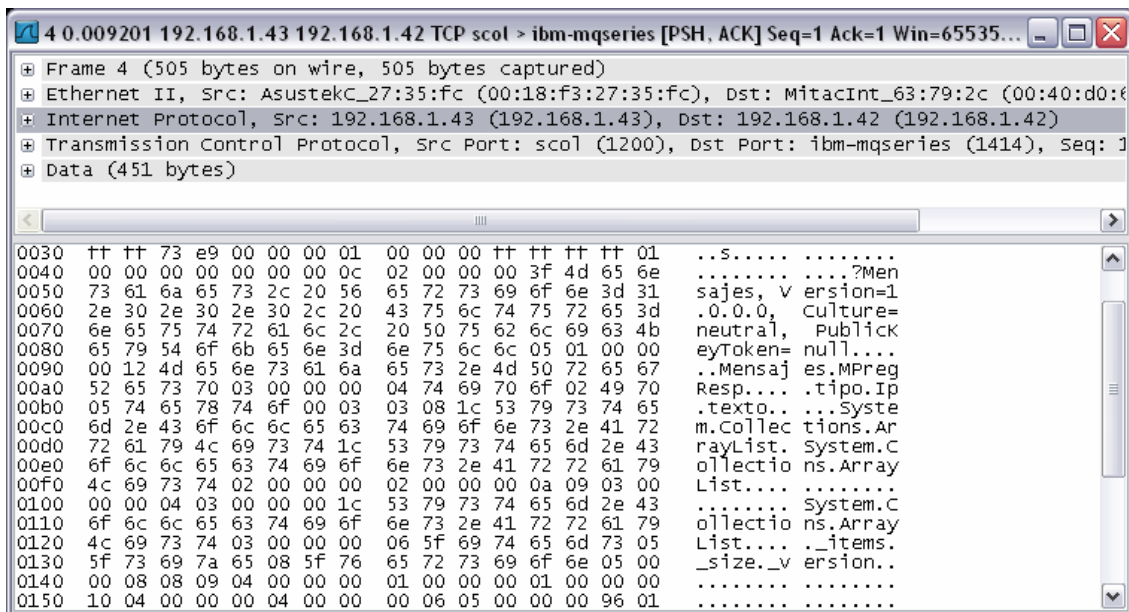


Fig. 4.2 Captura de un paquete de búsqueda con el programa Wireshark

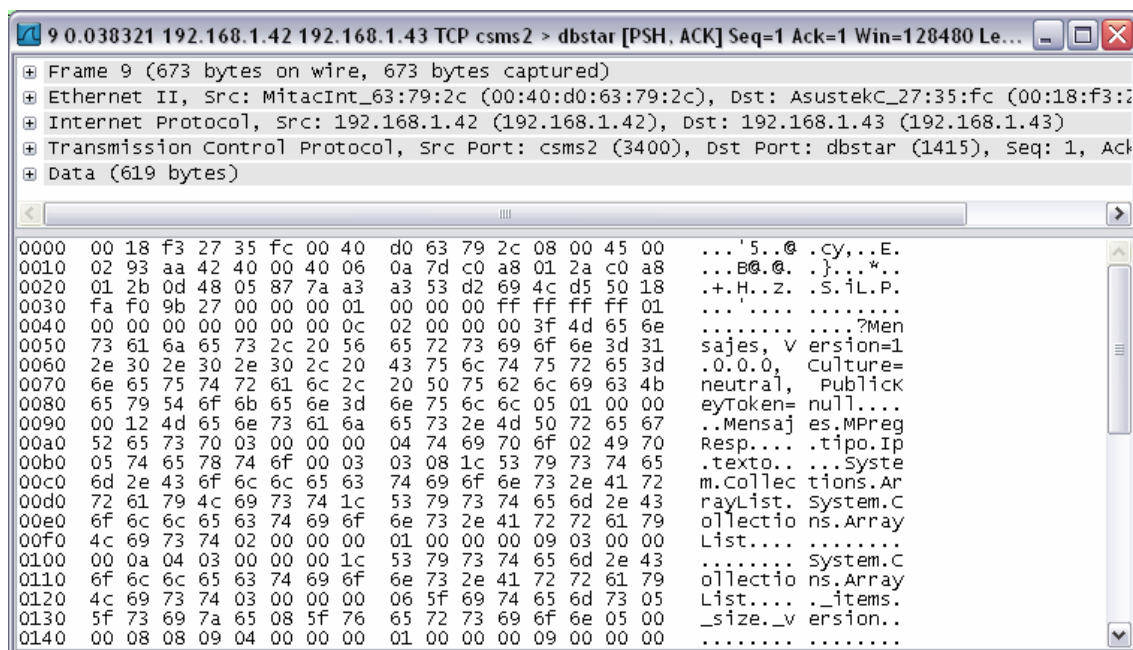


Fig. 4.3 Captura de un paquete de respuesta con el programa Wireshark

Tabla 4.1. Tipo y tamaño de los paquetes

	Tipo	Puerto destino	Tamaño
Anuncio	Muticast UDP	1413	720 bytes
Búsqueda	Unicast TCP	1414	505 bytes
Respuesta	Unicast TCP	1415	673 Bytes

4.2. Velocidad

El conjunto de pruebas siguientes, nos servirán para conocer mejor las posibilidades del sistema implementado, ya que conoceremos distintos tiempos que tarda el servidor en realizar varias funciones típicas.

4.2.1 Según ordenador

Para esta prueba, se medirá los tiempos que tarda el servidor en hacer diversas funciones como inicializarse y cargar varios nombres identificadores desde un archivo de texto, hacer búsquedas etc. Todas estas pruebas se harán en diversos ordenadores distintos en cuanto a potencia.

Para cronometrar los distintos tiempos que se tarda en hacer cada función, se introducen solo para estas pruebas, controles de tiempo al principio de donde queremos empezar a cronometrar, y al final, para después calcular el intervalo de tiempo.

4.2.1.1. Hardware utilizado

Para la realización de estas pruebas se ha optado para utilizar dos ordenadores de características distintas, las cuales son las siguientes:

Tabla 4.1. Características del hardware utilizado

	PC1	PC2
Tipo de CPU	Pentium M	Centrino Duo
Velocidad de CPU	1,4 GHz	1,7 GHz
Memoria RAM	1 GB	2 GB
Disco duro	5.400 RPM ATA	5.400 RPM ATA
Sistema operativo	Windows XP Pr.	Windows XP H.

Como se observa en la tabla, los ordenadores que se utilizan son de gama media, y de pocos años de antigüedad. Se empieza con un Pentium M más antiguo y con menos memoria RAM, para subir después a un ordenador de una gama más alta.

4.2.1.2. Inicialización

En esta primera prueba, se cronometra lo que se tarda en abrir el servidor, inicializarse, y cargar un árbol de prueba desde un archivo. Este archivo contiene muchos nombres identificadores, con una estructura como el ejemplo que se puede encontrar en los anexos.

Después de realizar esta prueba, los resultados obtenidos han sido los siguientes:

Tabla 4.2. *Tiempos medios de inicialización*

PC1	300 ms
PC2	328 ms

4.2.1.3. Búsquedas

Ahora vamos a medir el tiempo que tarda el servidor en hacer búsquedas por el árbol. Los puntos entre los que se cronometrarán, serán el momento en que recibe la petición de búsqueda, y el momento en que se devuelve a otra función los resultados obtenidos.

Esta prueba se hará con distintos nombres identificadores, simples y complejos. Primero se hará la búsqueda de varios nombres identificadores simples, y se conocerá la media que tarda en hacer una búsqueda simple. Después se hará la misma prueba con nombres identificadores más complejos, en las que tenga que entrar en varias ramas para encontrar los resultados y compararlos, y también se hará la media de dichas búsquedas.

Los resultados que se han obtenido en esta batería de pruebas, se recogen en las tablas 4.3 y 4.4.

Tabla 4.3. *Tiempos medios de búsqueda de los identificadores simples*

PC1	<1 ms
PC2	<1 ms

Tabla 4.4. *Tiempos medios de búsqueda de identificadores complejos*

PC1	<1 ms
PC2	<1 ms

La misma prueba se realiza con búsquedas inversas, en las que se pregunta por una dirección IP, y se devuelve en formato texto todas las ramas en las que se encuentra dicho servicio con esa dirección IP. Los resultados para esta prueba son los de la tabla 4.5.

Tabla 4.5. *Tiempos medios de búsquedas inversas*

PC1	<1 ms
PC2	<1 ms

Debido a que se tarda un tiempo inferior a un milisegundo en hacer cada búsqueda y que al ser tiempos tan pequeños, estos no se muestran bien, para concretar mejor lo que dura una búsqueda, y la velocidad real de búsqueda, se ha hecho otro tipo de prueba.

Esta nueva prueba, trata de que al lanzar una búsqueda, el servidor la haga mil veces seguidas, para después calcular la media de una búsqueda. Los resultados obtenidos, se pueden ver en la tabla 4.6 y 4.7.

Tabla 4.6. *Tiempos medios de búsquedas desde identificadores*

	Tiempo 1000 búsquedas	Tiempo media búsqueda
PC1	370 ms	0,37 ms
PC2	296 ms	0,296 ms

Tabla 4.7. *Tiempos medios de búsqueda inversa*

	Tiempo 1000 búsquedas	Tiempo media búsqueda
PC1	1,3 s	1,3 ms
PC2	781 ms	0,781 ms

Se observa que mejorando el ordenador, la velocidad de búsqueda es bastante más rápida. Aunque la velocidad de búsqueda es muy alta, si se quisiera utilizar el sistema en un ámbito de gran utilización por parte de los clientes, sería conveniente un ordenador potente. En el caso que el sistema lo tuviéramos para pocos clientes, con sistemas con poca capacidad de procesado, iría perfecto.

4.2.1.4. *Guardado en archivo*

Una vez hechas las pruebas de inicialización y de búsquedas, la última prueba de este apartado es la de guardado en archivo. Esta prueba consistirá en medir el tiempo que tarda el servidor en guardar todas las ramas del árbol en un archivo de texto, junto con las IPs que contienen cada una de las ramas.

En esta prueba, el cronometro mide desde que el servidor recibe la orden de guardar, hasta que cierra el archivo guardado. Se realiza con dos árboles, uno mediano, que es el mismo utilizado en las pruebas anteriores, y otro bastante más grande, para ver las diferencias de tiempo.

El archivo desde el que se carga este nuevo árbol más grande que el anterior, tiene la misma estructura. Los resultados después de la realización de este test, han sido los que se muestran en la tabla 4.8.

Tabla 4.8. *Tiempos medios de guardado en archivo*

	Árbol mediano	Árbol grande
PC1	20 ms	65 ms
PC2	15 ms	46 ms

El tiempo de guardado del árbol en un archivo de texto, es muy pequeño, teniendo en cuenta que escribe en el archivo todas las ramas y sus IPs correspondientes, y hasta para un árbol con gran información, el tiempo es muy corto, ya que se mantiene en decenas de milisegundos.

4.2.2 Según tamaño del árbol

Ahora se procede a hacer diversas búsquedas variando el tamaño del árbol, desde un árbol muy pequeño, hasta un árbol con gran cantidad de información. Gracias a las siguientes pruebas, podremos ver como va afectando en la velocidad de búsqueda, la cantidad de información que contiene el árbol.

4.2.2.1. Búsquedas por nombre identificador

Procedemos a crear distintos arboles cada uno con distinta cantidad de información guardada. Se empieza por el árbol más simple, y se va aumentando hasta arboles mucho más complejos, con gran cantidad de información.

Como se ha visto antes que fijarse en una sola búsqueda no sirve, ya que el tiempo es menospreciable, realizaremos otra vez una misma búsqueda 1000 veces, y sacaremos la media de una búsqueda. El nombre identificador que utilizaremos para la búsqueda, será complejo, y con el texto siguiente:

[país = España [ciudad = Barcelona]]
[servicio = impresora [color = si]
[ppp = 600]]
[accesibilidad = privada [departamento = IT]
[departamento = RRHH]]

Como en esta prueba ya no nos interesa comparar la velocidad según el ordenador donde funcione el servidor, sino en función de la cantidad de información del árbol, solo vamos a utilizar un ordenador, que será el antes mencionado como PC2.

Tabla 4.9. *Tiempos de búsqueda por nombre, según tamaño del árbol*

Ramas totales	Tiempo 1000 búsquedas	Tiempo medio búsqueda
5	281 ms	0,28 ms
25	281 ms	0,28 ms
50	281 ms	0,28 ms
100	281 ms	0,28 ms
150	296 ms	0,29 ms
250	281 ms	0,28 ms
350	296 ms	0,29 ms
500	296 ms	0,29 ms
750	296 ms	0,29 ms
1000	296 ms	0,29 ms
1500	296 ms	0,29 ms
2000	296 ms	0,29 ms
2500	296 ms	0,29 ms
3000	296 ms	0,29 ms
4000	296 ms	0,29 ms
5000	296 ms	0,29 ms
7500	296 ms	0,29 ms
10000	296 ms	0,29 ms
12000	296 ms	0,29 ms

Gracias al sistema de búsquedas implementado para las búsquedas principales que se utilizaran la gran mayoría del tiempo en nuestro sistema, se consigue que aunque aumente muchísimo el tamaño del árbol, el tiempo que se tarda en hacer una búsqueda, no aumente.

Este es un dato muy importante, al haber conseguido que el servidor solo tenga que ir a las ramas que se le indican desde el nombre identificador, y no tener que ir a todas.

4.2.2.2. Búsquedas por IP

Este caso, hacemos lo mismo que en la prueba anterior, pero esta vez con búsquedas inversas preguntando una dirección IP. Se muestran los resultados en la tabla 4.10.

Tabla 4.10. *Tiempos de búsqueda por IP, según tamaño del árbol*

Ramas totales	Tiempo 1000 búsquedas	Tiempo medio búsqueda
5	109 ms	0,10 ms
25	156 ms	0,15 ms
50	203 ms	0,20 ms
100	280 ms	0,28 ms
150	259 ms	0,25 ms
250	468 ms	0,46 ms
350	593 ms	0,59 ms
500	703 ms	0,70 ms
750	765 ms	0,76 ms
1000	1,14 s	1,14 ms
1500	1,42 s	1,42 ms
2000	1,98 s	1,98 ms
2500	2,46 s	2,46 ms
3000	2,81 s	2,81 ms
4000	3,56 s	3,56 ms
5000	4,43 s	4,43 ms
7500	6,60 s	6,60 ms
10000	8,76 s	8,76 ms
12000	9,67 s	9,67 ms

Aquí si que vemos que a medida que la cantidad de información del árbol va aumentando, la velocidad de búsquedas inversas va disminuyendo, dado que tiene que recorrer todas las ramas.

Esto en verdad no influirá negativamente en el sistema, ya que aunque con cantidades enormes de información, que pocas veces se llegarán a tener, se mantiene la búsqueda por debajo de los 10 ms. Otra cosa a tener en cuenta, es que este tipo de búsqueda, no es el principal, y pocas veces se utilizará.

4.2.2.3. *Inicialización*

La siguiente prueba se centra en el tiempo que tarda la aplicación del servidor en inicializarse y cargar el árbol desde un archivo de texto, y los resultados obtenidos se muestran en la tabla 4.11.

Tabla 4.11. *Tiempos de inicialización, según tamaño del árbol*

Ramas totales	Tiempo medio inicialización
5	62 ms
25	62 ms
50	109 ms
100	250 ms
150	234 ms
250	203 ms
350	234 ms
500	265 ms
750	328 ms
1000	390 ms
1500	531 ms
2000	656 ms
2500	781 ms
3000	890 ms
4000	1,18 s
5000	1,42 s
7500	2,07 s
10000	2,71 s
12000	3,31 s

El tiempo medio de inicialización, se mantiene bajo, teniendo en cuenta que la aplicación carga todas las partes necesarias para funcionar, así como inicializa los sockets de red para escuchar los anuncios y las peticiones.

Como es normal, a medida que la cantidad de información necesaria para generar el árbol aumenta, el tiempo de inicialización, también aumenta, pero siempre manteniéndose debajo de unos límites aceptables.

4.2.2.4. Guardado de la información en archivo

La última de las pruebas, medirá el tiempo que tarda el servidor en guardar todas las ramas del árbol y todas las IPs que guardan sus listas, en un archivo. Poco a poco como en las últimas pruebas, iremos aumentando la cantidad de información que contiene el árbol, y obtendremos los resultados que se muestran en la tabla 4.12.

Tabla 4.12. *Tiempos de guardado, según tamaño del árbol*

Ramas totales	Tiempo medio de guardado
5	<10 ms
25	<10 ms
50	<10 ms
100	<10 ms
150	<10 ms
250	<10 ms
350	15 ms
500	15 ms
750	15 ms
1000	15 ms
1500	15 ms
2000	15 ms
2500	31 ms
3000	31 ms
4000	31 ms
5000	46 ms
7500	62 ms
10000	78 ms
12000	93 ms

Es sorprendente lo poco que tarda en servidor en generar este archivo con toda la información de las ramas del árbol, dado que tiene que ir una por una por cada rama. Esta alta velocidad, también se mantiene con un árbol formado por cantidades enormes de información.

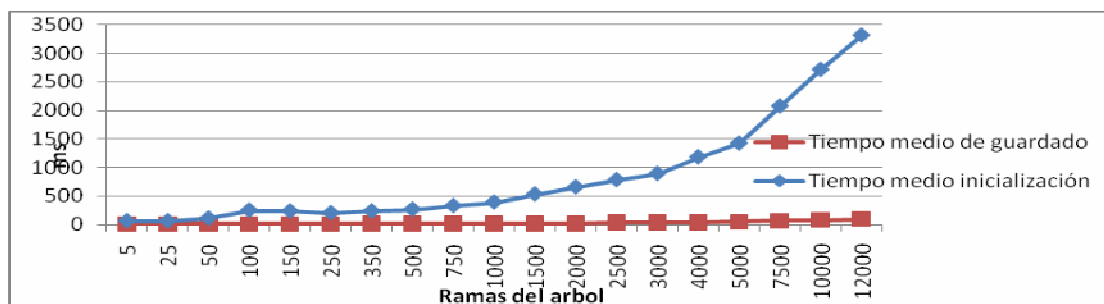
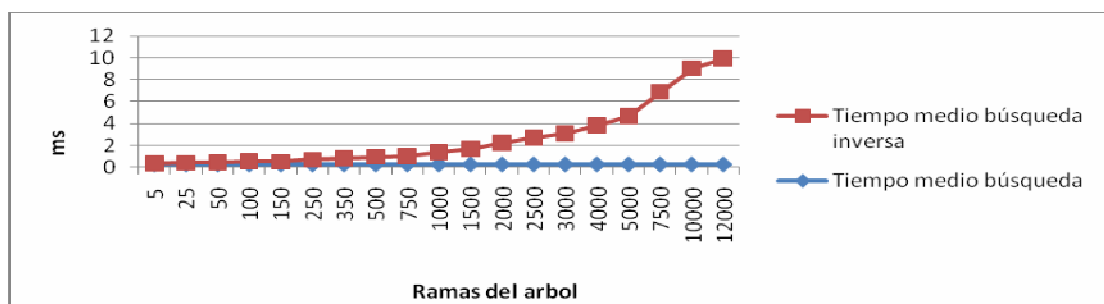


Fig 4.4 y Fig 4.5 *Gráficas de los resultados obtenidos en las pruebas de velocidad según cantidad de información en el árbol del servidor.*

CAPÍTULO 5. IMPACTO AMBIENTAL

Al tratarse de un proyecto básicamente de programación, el impacto ambiental de la aplicación, no existiría, ya que no afectaría de manera directa.

De manera indirecta, podemos tener en cuenta la energía consumida por los ordenadores donde correría la aplicación. En este ámbito, el impacto ambiental no es muy desfavorable, ya que al ser un sistema que consume pocos recursos, esta aplicación no necesitaría un nuevo ordenador específico para funcionar bien, sino que se podría utilizar en ordenadores que ya funcionan para otras aplicaciones

Además, comparado con otros sistemas, este sistema ayudaría a obtener más rápidamente los servicios que estamos buscando, al ser un sistema con búsquedas intencionales, lo que agilizaría los trabajos y bajaría el tiempo de utilización del ordenador, con la consiguiente reducción de gasto energético consumido en procesar.

Los ordenadores y componentes de hardware que van apareciendo en el mercado, cada vez son más ecológicos, tanto por los materiales usados, como en los gastos energéticos, pudiendo bajar la velocidad del reloj del procesador, cuando este no requiere funcionar al máximo, cosa muy favorable para el medio ambiente.

CAPÍTULO 6. CONCLUSIONES

Con la realización de este proyecto, se ha visto como la implantación de este sistema de nombres intencional, puede ser de gran utilidad, debido a su versatilidad, capacidad de personalización y velocidad.

La implantación de este sistema respecto a los sistemas habituales como los DNS, tiene muchas ventajas para los usuarios que quieran hacer búsquedas de servicios para así obtener por ejemplo su localización. Gracias a esto, los usuarios podrán formar el nombre a su gusto, con total libertad, escribiendo las características de los servicios que buscan, sin verse limitados a un nombre fijo como en los típicos sistemas de nombres.

El sistema no está limitado a un tipo de red concreta que abarque más o menos área, aunque su uso puede ser más útil en redes locales como oficinas, universidades, etc. Puede ser más útil en este tipo de redes, ya que servirá para tener una base de datos rápida y fiable, que puede funcionar en dispositivos de poca capacidad de procesamiento, y también gracias a esto, podría funcionar en ordenadores que ya usemos para otras cosas sin ver mermada su capacidad de procesamiento.

Después de hacer un gran número de pruebas, y guardar y buscar muchos nombres, se ha llegado a la conclusión de que cuando se guarde algo, se tiene que hacer con un poco de orden y con unas pequeñas bases. Es decir, por ejemplo los usuarios y servicios que vayan a usar el sistema, para que tener una buena utilidad, tienen que escribir los identificadores en un mismo idioma, ya que si alguien buscara con un identificador “[ciudad = barcelona]” y lo que hay guardado está en inglés “[city = barcelona]”, la búsqueda no devolvería ningún resultado. Otra base que se necesitaría saber para los usuarios, es si se utilizan minúsculas o mayúsculas, por el mismo problema.

Como se ha visto en el apartado de pruebas, un gran logro ha sido que las búsquedas a partir de un identificador, las cuales serán las que se utilizarán la gran mayoría de los casos, al estar orientado el sistema a estas búsquedas, estas no bajan su velocidad aunque el árbol aumente mucho de tamaño. Esto se ha logrado utilizando ciertas técnicas de programación para que el servidor no tenga que entrar en cada una de las ramas, sino que solo en las ramas necesarias. Gracias a esta característica, el servidor puede devolver las búsquedas muy rápidamente a la vez que puede contener mucha información, y muy bien organizada.

Tras finalizar el proyecto, y después de haber analizado la aplicación realizando gran cantidad de pruebas, se ve que se han logrado los objetivos de crear un sistema de nombres intencional, rápido, personalizable, y autoconfigurable. La programación mediante la plataforma .NET y lenguaje C#, al ser de nueva generación, ha logrado un código funcional, y con vistas de futuro para nuevas ampliaciones. Estas nuevas ampliaciones, podrían ser las de ver los servicios localizados en un mapa interactivo o implantar un traductor para los atributos.

BIBLIOGRAFIA

- [1] **Wikipedia. Domain Name System**
<http://es.wikipedia.org/wiki/DNS>
- [2] **INS: Intentional Naming System**
<http://nms.csail.mit.edu/projects/ins/>
- [3] **Microsoft Developer Network**
<http://msdn.microsoft.com/es-es/default.aspx>
- [4] **Página personal de Luis Gonzaga Pérez Córdón**
<http://www.di.ujaen.es/~lgonzaga/docencia.html>
- [5] **Devarticles**
<http://www.devarticles.com/c/b/C-Sharp/>
- [6] **C# Corner**
<http://www.c-sharpcorner.com/>
- [7] **C# Online**
[http://es.csharp-online.net/Main\(\)](http://es.csharp-online.net/Main())
- [8] **C# Online. TabControl**
<http://en.csharp-online.net/TabControl>
- [9] **Roy Roemaat, "Intentional Naming in Personal Networks"**
http://dacs.ewi.utwente.nl/news/archive/2004/files/abstract_roemaat.htm
- [10] **Scalability in an Intentional Naming System**
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.8460>



**Escola Politècnica Superior
de Castelldefels**

UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANEXOS

TITULO DEL TFC: Implementación de un sistema de direccionamiento intencional

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Antoni Carenys Roca

DIRECTOR: Juan López Rubio

FECHA: 4 de junio de 2009

ANEXO A. Ejemplo de archivo donde el servidor guarda el árbol

03/06/2009 22:44:15

[pais = Espana [ciudad = Barcelona [calle = Diputacio]]]

192.168.1.2

[pais = Espana [ciudad = Barcelona [calle = Serrano]]]

192.168.1.3

[pais = Espana [ciudad = Barcelona [calle = Aribau]]]

192.168.1.4

[pais = Espana [ciudad = Barcelona [calle = Major]]]

192.168.1.5

[pais = Espana [ciudad = Barcelona [calle = Verdaguer]]]

192.168.1.6

[pais = Espana [ciudad = Barcelona [calle = Sants]]]

192.168.1.7

[pais = Espana [ciudad = Barcelona [calle = Rambles]]]

192.168.1.8

[pais = Espana [ciudad = Barcelona [calle = Mait]]]

192.168.1.9

[pais = Espana [ciudad = Barcelona [calle = Rafael]]]

192.168.1.10

[pais = Espana [ciudad = Barcelona [calle = Joen]]]

192.168.1.11

[pais = Espana [ciudad = Barcelona [calle = Ferran]]]

192.168.1.12

[pais = Espana [ciudad = Barcelona [calle = Corts]]]

192.168.1.13

[pais = Espana [ciudad = Barcelona [calle = Migjorn]]]

192.168.1.14

[pais = Espana [ciudad = barcelona [calle = Bruc]]]

192.168.1.15

[pais = Espana [ciudad = Barcelona [calle = Pompeu]]]

192.168.1.16

[pais = Espana [ciudad = Barcelona [calle = Naim]]]

192.168.1.17

[pais = Espana [ciudad = Barcelona [calle = Moix]]]

192.168.1.18

[pais = Espana [ciudad = Barcelona [calle = Pensa]]]

192.168.1.19

[pais = Espana [ciudad = Barcelona [calle = Joc]]]

192.168.1.20

[pais = Espana [ciudad = Barcelona [calle = Gran Via]]]

192.168.1.21

[pais = Espana [ciudad = Barcelona [calle = Urquinaona]]]

192.168.1.22

[pais = Espana [ciudad = Barcelona [calle = Cerda]]]

192.168.1.23

-FIN-