Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Arquitectura de distribución adaptable para sistemas colaborativos implantados en la Web

Tesis que presenta

Javier Solís Angulo

para obtener el Grado de

Maestro en Ciencias

en Computación

Director de la Tesis

Dra. Sonia Guadalupe Mendoza Chapa

México, D.F. Enero 2009

Resumen

Las aplicaciones cooperativas distribuidas Web necesitan funciones dedicadas de compartición y de replicación de la información adaptadas a sus requisitos específicos.

En el presente trabajo de tesis se diseñó e implementó una arquitectura de distribución de entidades compartidas adaptable a diversos factores, e.g. capacidades de los dispositivos de almacenamiento y de procesamiento, derechos de acceso de los colaboradores sobre las entidades y decisiones de los colaboradores de disponer localmente de una copia (completa ó parcial) de una entidad. Para poder realizar dicho trabajo se estudiaron varios temas:

En primer lugar se estudian y se analizan las arquitecturas de distribución que permiten implantar sistemas colaborativos en la Web. Una arquitectura de distribución define la repartición de las entidades y de los componentes de un sistema colaborativo entre los sitios implicados en un proceso de colaboración. Principalmente, las arquitecturas de distribución difieren en tres aspectos: 1) la representación de cada entidad compartida en los sitios cooperantes; 2) el número de ejemplares de esta entidad presentes en el sistema y 3) la posible movilidad de una entidad entre los sitios participantes. Estos aspectos de la distribución pueden influenciar la forma de trabajo adoptada por los colaboradores, e.g. cada sitio participante que administra una replica de una entidad compartida permite a los colaboradores de trabajar de una manera autónoma. Sin embargo, dichos aspectos presentan diversas limitaciones de adaptación que dificultan la implantación de sistemas colaborativos en la Web, a causa de sus características como pueden ser, fallas potenciales de redes/servidores y retardos de comunicación.

En segundo lugar, se utiliza y se extiende el protocolo HTTP de la Web como soporte a la distribución de las entidades compartidas adaptadas. Por tal motivo se estudian sus aspectos y características más relevantes.

Desde el punto de la evaluación de las capacidades de los sitios cooperantes, el concepto de adaptabilidad de las entidades compartidas se realiza mediante técnicas de lógica difusa, tomando como referencia el esquema de los sistemas como Sugeno [Berzal, 2002]. Se utiliza el concepto de árbol de decisión para generar las reglas de inferencia. Dichas reglas sirven para determinar las entidades compartidas más adecuadas a ser enviadas al sitio participante.

Finalmente, otro aspecto relevante que se analiza en este trabajo es el principio de autor multi-sitios. Dicho principio es una guía de diseño de la plataforma PIÑAS que permite y facilita la cooperación entre personas físicamente distribuidas y potencialmente móviles. Más específicamente el principio distingue a los sitios participantes en una sesión de colaboración como un lugar de almacenamiento y otro de tratamiento de las entidades compartidas.

Palabras clave: Trabajo cooperativo asistido por computadora, sistemas colaborativos, arquitecturas de distribución, entidades compartidas, adaptación a dispositivos, lógica difusa.

Abstract

The Web distributed cooperative applications need suited functions for sharing and replicating information that fit their special requirements.

In this thesis work, it has been designed and implemented a distribution architecture of shared entities that is adaptable to different factors, e.g., capabilities of the storage and processing devices collaborator's access rights over the entities and collaborator's decision to get a (complete or partial) copy of an entity in order to access it locally. To perform this work several topics were studied.

First, the distribution architectures that allow the implementation of collaborative systems on the Web were studied and analyzed. A distribution architecture defines the replication policy of the entities and collaboration system components among the sites involved in a collaboration process. The distributions architectures mainly differ according to three aspects: 1) the representation of each shared entity in the cooperative sites, 2) the number of entity copies that exist in the system and, 3) the possibility for an entity to move among the involved sites. These distribution aspects can influence the way by which the collaborators work, e.g., each participating site that manages a copy of a given shared entity allows collaborators to work in an autonomous way. However, these aspects have different limitations of adaptation that make difficult the implantation of collaborative systems on the Web. These limitations directly come from characteristics such as network/server potential failures and/or communication delays.

Second, the Web HTTP protocol is used and extended in order to support the distribution of shared adaptable entities. For that reason the most relevant characteristics and aspects of this technology were studied.

From the evaluation of the cooperative site capabilities, the adaptability concept of the shared entities is realized using fuzzy logic techniques, taking as reference the schema of systems like Sugeno [Berzal, 2002]. The concept of decision tree is used for generating the inference rules that are used to determine the most suitable shared entities to be sent to the participating site.

Finally, another relevant aspect analyzed in this work is the multisite author concept. This main concept is a PIÑAS platform design guide that allows and eases the cooperation among physically distributed and potentially mobile people. Specifically this concept distinguishes the participating sites of each user collaboration session as a) a storage site and b) a site to apply treatments on the shared entities.

Keywords: computer supported cooperative work, collaborative systems, distribution architectures, shared entities, device based adaptation, fuzzy logic.

Agradezco al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional y al Consejo Nacional de Ciencia y Tecnología por el apoyo recibido en el transcurso de esta etapa de mi vida y gracias al cual pude alcanzar una meta más.

***A la Dra. Sonia**:*
Gracias por darme la oportunidad de trabajar con usted y de ir adquiriendo los conocimientos necesarios que serán muy útiles en mi vida profesional. Principalmente le agradezco por su apoyo incondicional en los momentos difíciles que se presentaron durante el desarrollo de la tesis. Muchas gracias.

***Al Dr. Dominique**:*
Le agradezco el tiempo que dedico para ayudarme a seguir avanzando en mi proyecto de tesis.

***A mis padres y hermanos**:*
Gracias por su apoyo constante e incondicional, ya que sin ustedes no hubiera logrado alcanzar esta meta. Esta tesis va dedicada a ustedes.

***A mis amigos y compañeros**:*
Gracias a todos mis compañeros y amigos que siempre me apoyaron.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto de investigación

La presente tesis de maestría se inscribe en el campo de investigación denominado "Trabajo Colaborativo Asistido por Computadora" o TCAC (CSCW en inglés). Este campo se enfoca tanto en los aspectos sociológicos de las actividades individuales y colectivas, como en los aspectos tecnológicos de la informática y de las comunicaciones para permitir a los miembros de un grupo colaborar eficientemente. Respecto al carácter multidisciplinario del TCAC, esta tesis toma principalmente la perspectiva informática (i.e., el desarrollo de sistemas colaborativos) con el objeto de estudiar las arquitecturas que permiten utilizar estos sistemas en entornos distribuidos (ver Figura 1.1).

Una arquitectura de distribución define la forma en la que las entidades compartidas (e.g. recursos multimedia) y los componentes de tratamiento de un sistema colaborativo están distribuidas en los sitios implicados en un proceso de colaboración. En particular, una arquitectura de distribución de entidades compartidas está caracterizada por tres aspectos [Dourish, 1998]: 1) la representación de la entidad en el sitio local, e.g. réplica o substituto (*proxy*); 2) el número de ejemplares (i.e. en forma de réplica) disponibles en el sistema, e.g. un solo ejemplar o varias réplicas de una entidad; y 3) la movilidad de la entidad entre los sitios participantes, e.g. entidad estática o dinámica.

Estos aspectos de la distribución de entidades pueden influenciar la forma de trabajo adoptada por los colaboradores, e.g. cuando una replica de una entidad compartida existe de manera permanente en cada sitio participante, los colaboradores pueden adoptar una forma de trabajo concurrente según la cual cada colaborador controla y modifica, de manera autónoma y simultánea, una parte diferente de la entidad compartida. Esta arquitectura de distribución, denominada "arquitectura totalmente replicada", ofrece un buen tiempo de respuesta de las acciones propias a nivel de la interfaz de usuario. Sin embargo, cuando los colaboradores están dispersos, esta arquitectura puede presentar retardos en la realimentación de las acciones

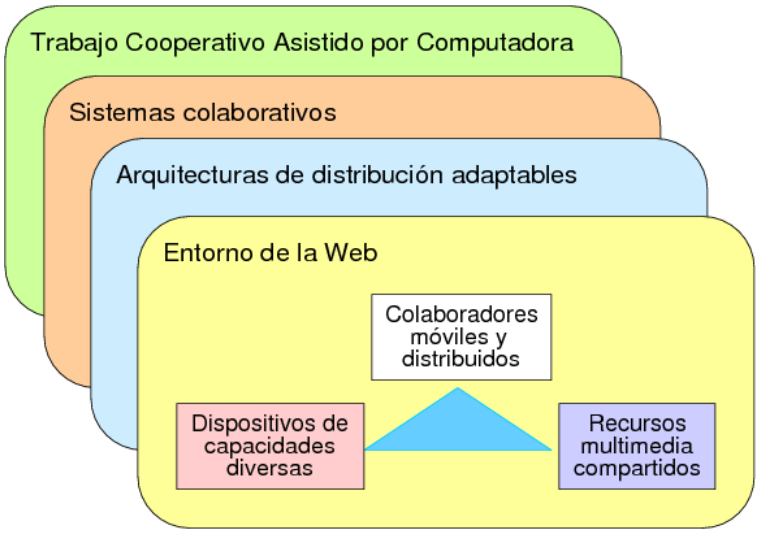

Figura 1.1: Contexto de investigación

de otros debido a las latencias de comunicación en Internet.

Este trabajo de tesis de maestría se centra precisamente en el estudio de arquitecturas de distribución diseñadas para implantar sistemas colaborativos que soporten personas geográficamente distribuidas. El diseño de arquitecturas de distribución en el entorno de una WAN lanza desafíos importantes a los desarrolladores de sistemas colaborativos, ya que este entorno presenta problemas que son omisibles en el entorno de una LAN, tales como la heterogeneidad de software y hardware, las fallas potenciales de redes y servidores y los retardos de comunicación.

Respecto al problema de la heterogeneidad, la tecnología Web ofrece una solución adaptada para ocultarla a nivel de [Bentley et. al., 1997]: 1) la plataforma, i.e. máquina, sistema operativo y red; 2) la interfaz de interacción, que es homogénea para las plataformas más utilizadas y 3) los lenguajes y soportes de programación, e.g. Java, que son independientes del sistema operativo y que permiten el desarrollo y la implantación eficaz de prototipos. Además, la tecnología Web facilita la integración de nuevos elementos a aplicaciones y datos Web existentes gracias a: 1) la utilización de estándares de facto para construir elementos Web, e.g. herramientas gráficas del dominio público con amplia difusión; 2) la definición de estándares públicos y simples, e.g. los lenguajes HTML y XML, el esquema de designación URL y el protocolo de comunicación HTTP; y 3) la evolución tecnológica que prosigue independientemente de fronteras organizacionales.

El advenimiento de dispositivos móviles, e.g. computadoras portátiles (*laptops*), organizadores personales (PDA) y teléfonos celulares (en particular, *smartphones*), hace ineludible la necesidad de considerar la movilidad en el diseño de arquitecturas de distribución para sistemas colaborativos, por dos razones principales [Mendoza, 2006]:

- La utilización de dispositivos móviles vuelve a los colaboradores potencialmente nómadas. En consecuencia, el acceso transparente a las entidades compartidas desde cualquier lugar y en cualquier momento, así como la localización y descarga automática de las entidades compartidas a partir del sitio más cercano al dispositivo de interacción del usuario, se vuelven requerimientos importantes en el diseño de arquitecturas de distribución.

- Las capacidades de almacenamiento, procesamiento, despliegue y comunicación de los dispositivos de interacción varían desde las potencialmente suficientes de una computadora portátil hasta las relativamente limitadas de un teléfono de tipo *smartphone*. Esta variabilidad de características impacta inevitablemente la distribución de entidades compartidas en términos de representación, número de ejemplares y movilidad. Por lo tanto, una arquitectura de distribución implantada en un tal entorno debería ser capaz de adaptarse a esa variabilidad.

1.2. Planteamiento del problema

Como ya se mencionó en la sección precedente, las arquitecturas de distribución difieren principalmente en tres aspectos: 1) la representación de una entidad compartida en el sitio local; 2) el número de ejemplares de una entidad presentes en el sistema y 3) la movilidad de una entidad entre los sitios participantes. Estos aspectos de la distribución dificultan la implantación de sistemas colaborativos en el entorno de la Web porque pueden tener repercusiones desfavorables en el proceso de colaboración. Algunos ejemplos de dichas repercusiones son:

1. si el dispositivo receptor no dispone de las capacidades suficientes para recibir, almacenar y procesar una entidad, entonces recibirá (según principio de todo o nada) un representante de la entidad, el cual se limita a proveer metadatos, e.g., nombre, ubicación lógica en el espacio de trabajo, dimensiones y formato;

2. la existencia de un solo ejemplar de una entidad podría: a) incrementar el tiempo de respuesta a medida que crece el número de participantes en una sesión, b) requerir un alto consumo de ancho de banda y c) no garantizar la disponibilidad de las entidades en un entorno sujeto a fallas potenciales de red y de servidores;

3. incremento del espacio de almacenamiento requerido por el hecho de guardar una réplica de cada recurso multimedia en cada sitio participante, aún cuando algunos colaboradores no requieran algunos de estos recursos o no tenga derecho de accederlos.

Las desventajas de tales aspectos de la distribución de entidades se ponen en evidencia a continuación.

1.2.1. Representación de una entidad compartida

El primer aspecto que influye en las arquitecturas de distribución se refiere a la representación de las entidades compartidas [Mendoza, 2006]. Una entidad compartida que reside en un sitio único se comporta como un servidor, i.e., de manera transparente puede recibir y procesar solicitudes de acciones ("mensaje") provenientes de otros sitios. La transparencia relativa al envío de un mensaje permite a una entidad solicitar un procesamiento sobre otra entidad sin tener que preocuparse por saber si esta entidad destinataria es local o distante. Con el fin de lograr tal transparencia, cada entidad es representada localmente por un substituto. De esta manera, cuando una entidad envía un mensaje a una entidad distante, el mensaje primero es expedido al substituto local, el cual lo reexpide a la entidad real distante que representa. La entidad distante entonces realiza el procesamiento solicitado y finalmente, a través del substituto, responde al solicitante sin necesidad de saber si está localizado en un sitio diferente.

La utilización de substitutos no es siempre el enfoque más eficiente para distribuir una entidad compartida. De manera alternativa, la duplicación de una entidad remota en el sitio local puede aumentar significativamente la prestación (*performance*) global del sistema colaborativo. Por ejemplo, la lectura de una cadena de caracteres localizada en un sitio distante, debiéndose efectuar según una política que obtiene un carácter a la vez, necesitará la transmisión de un gran número de mensajes. Por lo tanto, la duplicación de esta entidad permitiría aumentar sensiblemente la eficiencia de esta operación.

Así, desde el punto de vista meramente técnico, la representación de una entidad compartida en forma de substituto permite crear representantes en nuevos sitios participantes, con el fin de poder referenciar de manera transparente la entidad remota (técnica empleada en las arquitecturas denominadas "centralizadas"). Por el contrario, la representación de una entidad compartida en forma de réplica permite crear ejemplares en nuevos sitios participantes, con el fin de poder referenciar directamente la entidad local (técnica empleada en las arquitecturas denominadas "replicadas").

Con el advenimiento de dispositivos de baja capacidad, e.g., PDAs y *smartphones*, la representación binaria de una entidad (i.e., como réplica –todo o como substituto –nada) en el dispositivo de recepción, no es suficiente. Para incentivar la colaboración entre personas distribuidas, sus entidades compartidas también deberían ser accesibles (tanto para consulta como para modificación) mediante estos dispositivos. Por lo tanto, la arquitectura de distribución de un sistema colaborativo debería proveer un mecanismo para adaptar los recursos multimedia al dispositivo de recepción. No obstante, ninguna de las infraestructuras para el desarrollo de sistemas colaborativos, que han sido propuestas hoy en día, ofrece tales mecanismos de adaptación.

1.2.2. Número de ejemplares en el sistema

El segundo aspecto que influencia las arquitecturas de distribución se refiere al número de ejemplares de una entidad que están disponibles en un sistema colabora-

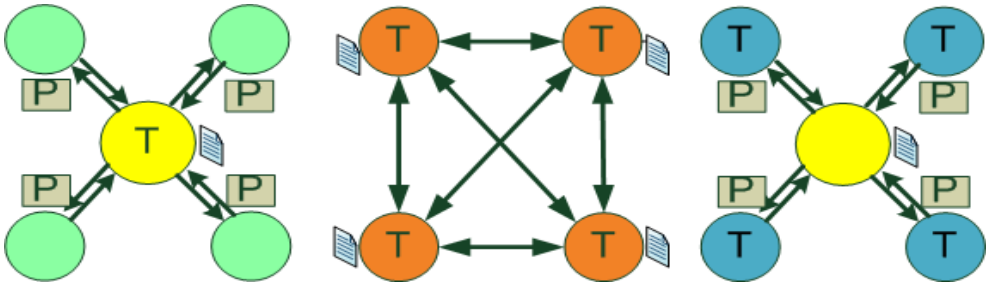

Figura 1.2: Arquitecturas centralizada (a), replicada (b), híbrida (c)

tivo. Este aspecto está relacionado con la representación de la entidad compartida, pero es relativo a la cantidad de sitios disponibles a un grupo de colaboradores. Para poner en evidencia este relativismo, suponga que n representa el número de sitios disponibles a un grupo de colaboradores, r es el número de ejemplares de una entidad y p es el número de *proxies* de la entidad:

Una arquitectura sigue un esquema centralizado cuando existe un sitio conocido que almacena un solo ejemplar de toda entidad del sistema. El sitio de cada colaborador almacena un *proxy* de cada entidad, cuyo objetivo es ocultar el carácter remoto del único ejemplar. Generalmente, el sitio centralizado no actúa como sitio de colaborador. En consecuencia, toda entidad del sistema tiene la propiedad invariante siguiente: $r = 1$ y $p = n - 1$. Como se ilustra en la figura 1.2a, por cada entidad del sistema, existe un solo ejemplar localizado en el sitio centralizado y cuatro *proxies* repartidos equitativamente entre los cuatro sitios de colaborador.

Por el contrario, una arquitectura sigue un esquema replicado cuando cada sitio de colaborador almacena una replica de toda entidad del sistema. Por lo tanto, el concepto de *proxy* se vuelve prácticamente irrelevante, ya que cada sitio de colaborador puede acceder localmente a las entidades del sistema. A diferencia del esquema anterior que hace intervenir sitios que juegan roles diferentes, la arquitectura replicada involucra únicamente sitios pares, i.e., que tienen el mismo rol. Así, cada entidad del sistema posee la siguiente propiedad invariante: $r = n$, y $p = 0$. Tal como se muestra en la figura 1.2b, cada uno de los cuatro sitios de colaborador almacena una replica de cada entidad del sistema.

Una arquitectura híbrida combina los dos esquemas precedentes: algunas entidades del sistema siguen un esquema centralizado, mientras que las entidades restantes siguen un esquema replicado. Así la arquitectura híbrida hace intervenir tanto un sitio centralizado como varios sitios pares. Como se ilustra en la figura 1.2c, el sitio centralizado almacena un ejemplar de algunas entidades del sistema, mientras que cada uno de los cuatro sitios de colaborador guarda un *proxy* de estas entidades centralizadas así como una réplica de cada una de las entidades restantes del sistema.

La arquitectura centralizada facilita la implementación de algunas funciones de administración, e.g. persistencia y coherencia de las entidades compartidas debido a la existencia de un solo ejemplar de éstas [O'Grady, 1996] [Lukosch, 2003]. Sin embargo,

esta arquitectura está caracterizada por su fuerte consumo de ancho de banda y por su sensibilidad a latencias de red, ya que la comunicación entre el sitio servidor y los sitios cliente se realiza por medio de eventos. Aunque esta arquitectura podría tener un desempeño relativamente aceptable en una LAN de alta velocidad, la existencia de un servidor único constituye la principal restricción para transponerla en el entorno no confiable de Internet, ya que las fallas potenciales en conexiones de red no permiten asegurar en todo momento la disponibilidad de las entidades compartidas.

En contraste con el esquema precedente, la arquitectura replicada ofrece una ventaja en tiempo de respuesta en la interfaz de usuario, ya que el acceso a las entidades se efectúa de manera local. Otra ventaja importante que la designa como una arquitectura adecuada para el entorno no confiable del Internet reside en la imposibilidad de cuellos de botella, gracias a la descentralización tanto de los componentes de tratamiento como del almacenamiento de las entidades compartidas. Sin embargo, la constante evolución de la Web ha engendrado una diversidad de recursos multimedia, e.g. texto, imágenes, audio y video, que se revelan demandantes en espacio de almacenamiento. En consecuencia, resultaría muy costoso (en términos de espacio de almacenamiento requerido) implantar una arquitectura replicada generalizada en el entorno de la Web.

La arquitectura híbrida combina las ventajas de los esquemas centralizado y replicado, sin embargo también recupera sus desventajas.

1.2.3. Movilidad de la entidad

El tercer aspecto que influye las arquitecturas de distribución, se refiere a la movilidad de una entidad entre los sitios participantes. La mayoría de las arquitecturas de distribución, e.g. [Phillips, 1999] [Roth y Unger, 2000], consideran que las entidades compartidas se utilizan (tanto en consultación como en modificación) en el mismo lugar donde están almacenadas de manera persistente. Sin embargo, el advenimiento de la tecnología Web ha lanzado nuevos retos a los desarrolladores de sistemas colaborativos en este sentido.

La arquitectura cliente/servidor de la Web almacena tanto entidades (e.g. recursos multimedia) como componentes de tratamiento (e.g. *applets*) en el sitio del servidor y los migra a los sitios cliente donde serán respectivamente utilizados y ejecutados. Este es un ejemplo de una de las estrategias más utilizadas en la Web, pero seguramente no es la más adecuada. Esta estrategia de distribución igualmente ha sido empleada por WebDAV [Goland et. al ., 1999] y BSCW [Appelt, 2001] para soportar la producción colaborativa de documentos en la Web. Sin embargo, en el marco de entornos de colaboración no confiables, como el Internet, esta estrategia limita particularmente la disponibilidad de las entidades compartidas, debido a que la ocurrencia de fallas de red y de los sitios servidores es relativamente elevada.

Fuera del entorno de la Web, algunas infraestructuras colaborativas proponen soportes de distribución más robustos para dicho entorno no confiable. A pensar de los elementos de solución que puedan ser propuestos, estas infraestructuras no

toman en consideración un parámetro de distribución que es esencial para extender la arquitectura de la Web: la distinción entre el lugar de almacenamiento de las entidades compartidas y de los componentes de tratamiento y sus respectivos lugares de utilización y de ejecución.

Además, dado que estas infraestructuras sólo ofrecen soporte para el trabajo colaborativo en modo síncrono (i.e., los colaboradores convergen simultáneamente en una sesión), no abordan otros aspectos esenciales como la disposición de los sitios concernientes, los derechos de acceso de los colaboradores, las capacidades de almacenamiento, de procesamiento, de despliegue y de comunicación de los dispositivos de recepción y la propia decisión del usuario. Estos aspectos resultan directamente de la necesidad de soportar el trabajo en modo desconectado o degradado, requerimiento ineludible de la interacción del usuario en el entorno de la Web.

1.3. Objetivos del proyecto

1.3.1. Objetivo general

Diseñar e implementar una arquitectura de distribución adaptable en la Web que ponga las entidades compartidas requeridas a la disposición de los usuarios, con el fin de lograr una colaboración efectiva durante una sesión de trabajo colectiva o individual. La administración de la adaptación y de la distribución de las entidades compartidas se lleva a cabo respectivamente mediante la aplicación de la lógica difusa y el uso de la tecnología *peer to peer*.

1.3.2. Objetivos específicos

1. Desarrollar un servicio para adaptar la representación de una entidad compartida, en los sitios de recepción, en función de los siguientes aspectos: a) la disposición en Internet de los sitios participantes, b) los derechos de acceso de los colaboradores, c) las capacidades de almacenamiento y procesamiento de los dispositivos de recepción y d) la propia decisión del colaborador.

2. Definir un esquema de distribución que garantice a los colaboradores la disponibilidad de sus entidades compartidas en un entorno con fallas latentes de red y de servidores.

3. Desarrollar un mecanismo de migración de entidades que se integre con la arquitectura cliente/servidor de la Web, en términos de la distinción entre almacenamiento y uso de una entidad compartida.

1.4. Organización de la tesis

Esta tesis está organizada en ocho apartados: del segundo al cuarto capítulo se expone el marco teórico de las arquitecturas de distribución, de la lógica difusa, de los árboles de decisión y de los mecanismos de comunicación. En los dos siguientes capítulos se explica el diseño del sistema propuesto, así como las pruebas y los resultados obtenidos. En el séptimo se exponen las conclusiones y el trabajo futuro y finalmente se presentan los anexos. A continuación se detalla cada uno de los capítulos que forman parte de esta tesis (ver figura 1.3).

Arquitecturas de distribución. En este capítulo se realiza un estudio de las principales arquitecturas de distribución implementadas en plataformas y sistemas colaborativos. También se estudian algunas infraestructuras relevantes que soportan un solo tipo de arquitectura de distribución no flexibles (no adaptables), así como algunas infraestructuras flexibles (adaptables) que soportan varios tipos. Finalmente se presenta una categorización de las entidades compartidas y de los mecanismos de tratamiento, de acuerdo al lugar donde son almacenados, utilizados y ejecutados.

Lógica difusa y árboles de decisiones. En este capítulo se describen varios conceptos relevantes en la teoría de conjuntos difusos: su representación gráfica, sus operaciones y sus relaciones difusas. Se ofrece una descripción formal de los sistemas basados en reglas, haciendo énfasis en los sistemas de tipo Mamdani y de tipo Sugeno. Por último, se describe la estructura de un árbol de decisión, así como el procedimiento para generar las reglas de inferencia utilizando las ramificaciones de dicho árbol.

Arquitecturas y mecanismos de comunicación. En este capítulo se realiza una clasificación de las arquitecturas de redes basadas en los modelos cliente-servidor y *peer to peer*. Se explica el funcionamiento del protocolo HTTP, así como su pertenencia a los modelos OSI y TCP/IP. Además se describen en general algunos servidores Web basados en código libre y en particular el servidor Web Apache. Asimismo, se presentan algunos de los servicios más importantes de Internet, el estándar MIME y la denominación de equipos de cómputo en Internet. Finalmente, se describe la metodología CGI y la evolución de los lenguajes utilizados en el desarrollo de aplicaciones Web.

Diseño del sistema. En este capítulo se explica el modelado del sistema propuesto mediante diagramas de flujo de datos, en el cual se cubren los siguientes aspectos: 1) el contexto general del sistema, 2) la decisión del usuario sobre los recursos Web que quiere recibir, 3) la obtención de las capacidades del dispositivo, 4) la determinación del recurso que será enviado al sitio participante y 5) la distribución de los recursos en los diferentes sitios de almacenamiento. Por último se expone la generación de las reglas de inferencia utilizando las ramificaciones del árbol de decisiones.

Implementación, pruebas y resultados. En este capítulo se explica la forma en que son obtenidas las capacidades del sitio participante, la adaptación de los recursos compartidos a los dispositivos aplicando el concepto de fuzzificación y el motor de inferencia diseñado especificamente para la arquitectura de distribución adaptable. Se expone la implementación de los módulos definidos en el capítulo del diseño del

sistema, haciendo hincapié en la obtención y el procesamiento de los datos estándares, datos de control y recursos Web. Por último se explica el funcionamiento del mecanismo de distribución.

Conclusiones y trabajo futuro. Aquí se exponen las conclusiones a las que se llegó después de haberse realizado el trabajo de tesis. Además se proponen algunas modificaciones y ampliaciones que se pueden llevar acabo en el futuro, siendo el enfoque principal los usuarios, las redes inalámbricas y dispositivos móviles.

Apéndices. Se mencionan algunos conceptos básicos de la lógica difusa, además de las tablas utilizadas para la generación de las reglas difusas. Por último se da una explicación general de la compilación, la instalación, la configuración y la ejecución del servidor Web Apache.

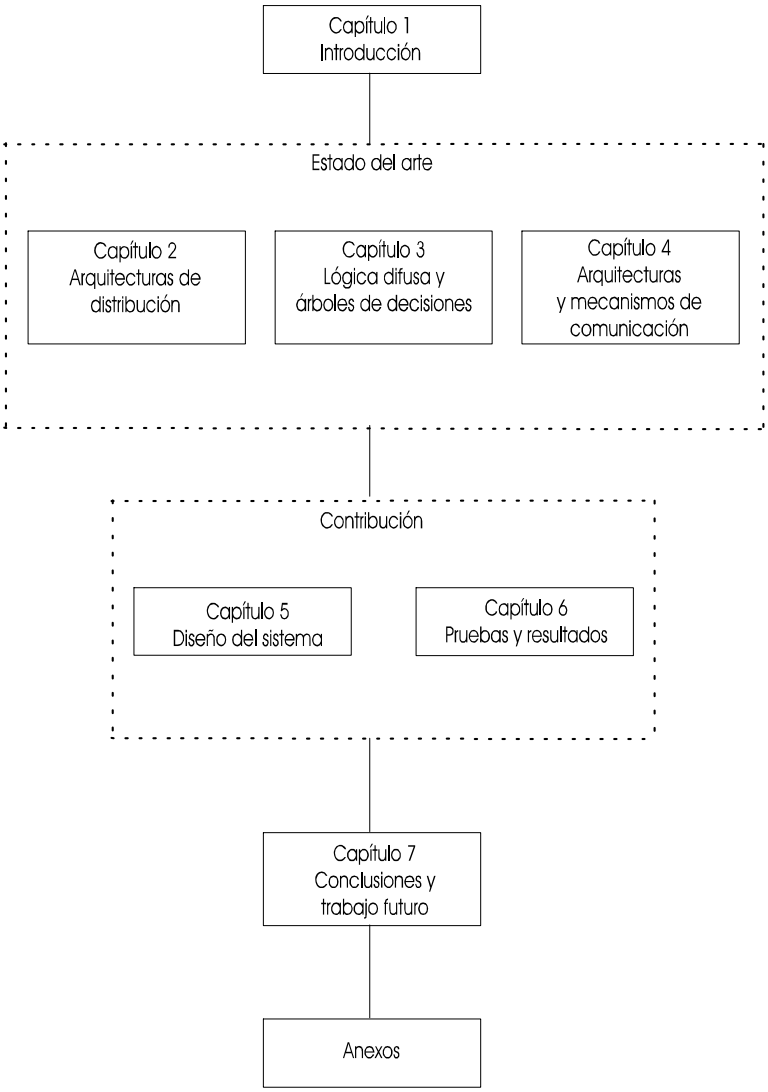

Figura 1.3: Organización de la tesis

Capítulo 2

Arquitecturas de distribución

En este capítulo se estudian algunas de las principales arquitecturas de distribución de los sistemas colaborativos. Dependiendo de su estructura y de sus características generales, las arquitecturas de distribución se clasifican en: centralizadas, replicadas, híbridas, asimétricas y de múltiples servidores. Dicha clasificación está basada en los esquemas de aplicación y distribución: el primero describe los componentes que integran el sistema colaborativo y la forma en que se efectúa la interacción entre ellos; el segundo esquema se refiere a la distribución de dichos componentes entre los sitios participantes. En base a esta clasificación, se analizan algunas infraestructuras que ofrecen servicios de distribución no adaptables y adaptables, las cuales soportan respectivamente uno y varios tipos de arquitecturas. Por último, se describe una categorización de los datos y del código ejecutable en la Web, dependiendo del lugar donde se almacenan, usan o ejecutan.

2.1. Clasificación de arquitecturas de distribución

El concepto de "trabajo colaborativo" involucra un grupo de personas geográficamente distribuidos, que realizan sus actividades sobre una plataforma de trabajo. Para permitir que los colaboradores puedan trabajar eficientemente, es necesario enfocarse en resolver los problemas latentes de este entorno distribuido [Roth y Unger, 2000], como pueden ser fallos en la red y en los servidores. Algunas consecuencias que traen consigo estos problemas son: el acceso a información obsoleta e incoherente y la indisponibilidad de recursos.

La transparencia de colaboración, la consciencia de colaboración, el tiempo de respuesta, así como la consistencia y persistencia son algunos factores importantes que deben tomarse en cuenta durante el diseño de la arquitectura de distribución de las entidades compartidas [Lukosh, 2003] [Roth y Unger, 2000]:

- **Transparencia de colaboración**: el proceso de distribución debe ser automático e imperceptible para el usuario, de tal forma que este no se involucre

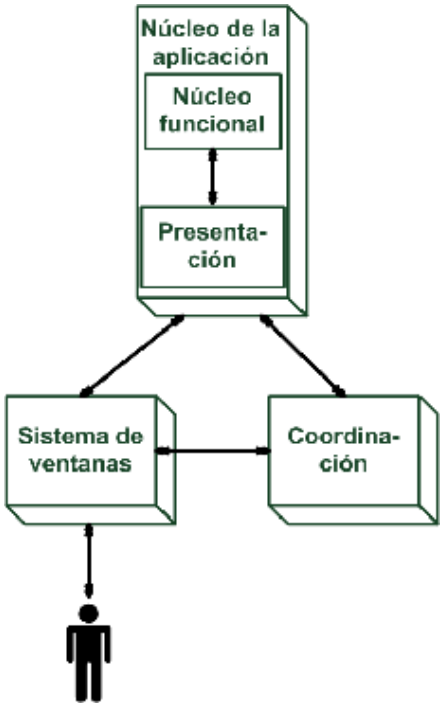

Figura 2.1: Esquema de aplicación

ni interactúe directamente con el mecanismo de distribución.

- **Consciencia de colaboración**: se refiere a que el usuario interactúa directamente con el mecanismo de distribución.
- **Tiempo de respuesta**: es el tiempo que una aplicación necesita para acceder a las entidades compartidas y propagar los posibles efectos a la interfaz de usuario.
- **Consistencia y persistencia**: concierne el aseguramiento de la coherencia y el manejo de la persistencia de las entidades compartidas.

La clasificación de las arquitecturas de distribución está basada en los esquemas de aplicación y de distribución. El **esquema de aplicación** define los componentes del sistema colaborativo y la forma en cómo interaccionan entre ellos. El **esquema de distribución** especifica cómo los componentes definidos en el esquema de aplicación están distribuidos entre los sitios participantes [Roth y Unger, 2000][Lukosh, 2003]. Dichos esquemas son explicados más a detalle en los siguientes apartados:

2.1.1. Esquema de aplicación

Como puede observarse en la figura 2.1, un sistema colaborativo está integrado por tres partes principales y dos secundarias [Roth y Unger, 2000], además de los enlaces entre los componentes, los cuales indican el flujo de datos:

1. **Núcleo de la aplicación:** representa la funcionalidad del sistema colaborativo, la cuál está formada por un núcleo funcional y un componente de presentación.

   - **Núcleo funcional:** es el conjunto de mecanismos encargados del tratamiento de las entidades compartidas (e.g., multimedia) sobre las cuales colaboran los usuarios de manera conjunta y organizada.
   - **Presentación:** son los mecanismos que trabajan sobre las entidades compartidas para su despliegue en pantalla.

2. **Sistema de ventanas:** permite el despliegue en ventanas de las entidades compartidas que fueron procesadas por el componente de presentación. Además recibe los eventos de los dispositivos periféricos (e.g., teclado y ratón) asociados al sitio.

3. **Coordinación:** es el mecanismo encargado de la gestión del sistema colaborativo en un entorno distribuido. Algunas tareas de coordinación son la sincronización de la entrada de comandos y el control de concurrencia a las entidades compartidas. Asimismo, puede requerirse un servicio de notificación que informe sobre las modificaciones efectuadas en las entidades compartidas [Lukosh, 2003].

2.1.2. Esquema de distribución

El esquema de distribución especifica cómo los componentes definidos en el esquema de aplicación están distribuidos entre los sitios participantes. Se pueden diferenciar cinco tipos básicos de arquitecturas de distribución a partir de los cuales se pueden generar otros más.

1. **Arquitecturas con componentes centralizados:** estas arquitecturas tienen al menos un componente centralizado sobre el cual trabaja un grupo de colaboradores. En consecuencia, cada sitio participante interactúa únicamente con el componente compartido ubicado en el servidor. Algunas arquitecturas de este tipo son: la aplicación centralizada, el estado centralizado y la coordinación centralizada.

   - **Aplicación centralizada:** existe un control entre usuarios, lo cual implica una colaboración consciente ya que, por cada usuario, el servidor mantiene un componente de presentación que genera las salidas. También existe un núcleo funcional que coordina dichos componentes, así cada cliente tiene una actualización privada [Phillips, 1999]. La principal desventaja de esta arquitectura es una gran carga en la red, ya que cada cliente tiene un sistema de ventanas que interactúa constantemente con el servidor. Por tanto, solo se utiliza en redes de área local (ver figura 2.2a).

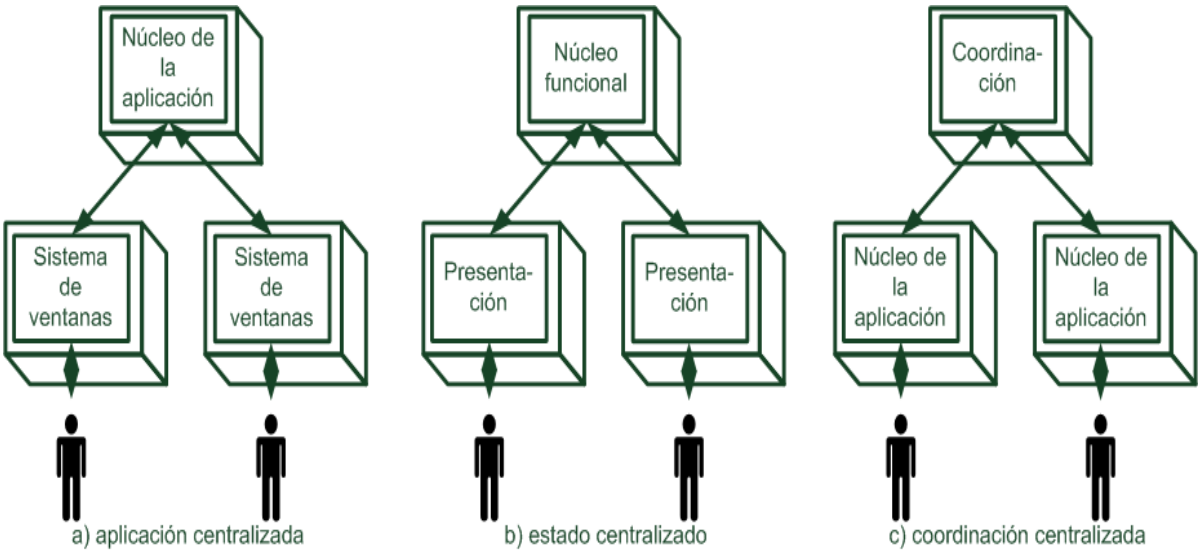

Figura 2.2: Arquitecturas con componentes centrales

- **Estado centralizado:** el núcleo funcional se encuentra en el servidor mientras que los componentes de presentación están descentralizados en los sitios participantes, con el fin de disminuir la cantidad de mensajes transferidos. La principal desventaja de esta arquitectura es la indisponibilidad de información en tiempo real (ver figura 2.2b).
- **Coordinación centralizada:** en el servidor solo se localiza el componente de coordinación, el cual asegura el mantenimiento de la consistencia y la secuencia de los eventos provenientes de los sitios participantes en donde se ubica el núcleo de la aplicación. La principal desventaja de esta arquitectura es la generación de cuellos de botella.

  Según Phillips [Phillips,1999] los sistemas conscientes de colaboración comparten muchas de las ventajas y desventajas de la **arquitectura de comunicación directa** (ver inciso 2) como la relativa simplicidad de implementar el mecanismo de consistencia mediante un algoritmo centralizado. Sin embargo, el mecanismo de consistencia radica en un solo servidor y si este fallara provocaría la caída del sistema. Además, el tiempo de respuesta en el sistema de ventanas es pobre en comparación con la arquitectura de comunicación directa (replicada), ya que la actualización de una entidad compartida requiere un mínimo de dos transmisiones en red (ver figura 2.2c).

2. **Arquitecturas con comunicación directa (replicada):** este tipo de arquitecturas no tiene componentes centralizados. Por el contrario, cada participante está conectado con todos los demás que forman parte del grupo, lo que resulta en una ejecución multi-sitio. Algunas arquitecturas de este tipo son: la apli-

cación descentralizada, el estado replicado y semi-replicado [Lukosh, 2003], el estado replicado con conexión de componentes de presentación y la coordinación descentralizada.

- **Aplicación descentralizada**: cada sitio participante dispone de los tres componentes del esquema de aplicación: el núcleo de la aplicación, el sistema de ventanas y la coordinación que se encarga de la conexión entre los sitios (ver figura 2.3a). Dicha distribución de los componentes soporta el trabajo desconectado.
- **Estado replicado:** el núcleo funcional está separado del componente de presentación, pero ambos son administrados por cada sitio participante. Esta arquitectura mejora el tiempo de respuesta en la interfaz de usuario, ya que las solicitudes son manejadas localmente. Además el núcleo funcional permite la comunicación a través de la red (ver figura 2.3b). Para asegurar la consistencia de las entidades compartidas se debe disponer de un mecanismo de sincronización y de control de concurrencia, cuyos algoritmos son complejos de desarrollar. El tráfico en la red se incrementa debido a que cualquier cambio realizado por un sitio participante sobre una entidad compartida es propagado a todos los demás [Lukosh, 2003].
- **Estado replicado con conexión de componentes de presentación:** los núcleos funcionales de cada sitio participante están vinculados entre ellos y, a su vez, cada uno de dichos núcleos está enlazado con el componente de presentación local. Además, los componentes de presentación de los sitios participantes están interconectados entre sí (ver figura 2.3c). Esta distribución eleva el número de mensajes en la red, ya que existen entidades compartidas que deben ser actualizadas constantemente e.g., el tele-puntero, cuyas coordenadas se transmiten a cada sitio participante en una sesión colaborativa.
- **Coordinación descentralizada:** una ventaja de esta arquitectura es que se evitan los cuellos de botella causados por el servidor de la arquitectura centralizada. Sin embargo, los mecanismos de control utilizados son más complejos, ya que la coordinación y el núcleo de la aplicación se encuentran en cada sitio participante (ver figura 2.3d). Los usuarios interaccionan directamente con estos mecanismos, lo que conlleva a una conciencia de colaboración.
- **Estado semi-replicado**: una variante del estado replicado es el estado semi-replicado en donde las entidades compartidas están distribuidas en más de un sitio de almacenamiento, pero no necesariamente en todos. La principal diferencia del estado semi-replicado con respecto a las demás arquitecturas, es el incremento en la disponibilidad de las entidades compartidas. Dichas entidades están distribuidas en varios sitios que han sido predefinidos para el almacenamiento. Sin embargo, también desempeñan

la función de sitio de trabajo, ya que contienen componentes del sistema de ventanas. Los demás sitios solo disponen del sistema de ventanas, el cual tiene la capacidad de acceder a las entidades compartidas que se encuentran remotamente.

Las arquitecturas semi-replicadas combinan los beneficios y las debilidades de las arquitecturas centralizadas y replicadas. Los sistemas semi-replicados conscientes de colaboración generalmente tienen una intensiva actividad de visualización y actualización en los sitios participantes. Si el protocolo entre el sitio participante y el sitio servidor es estandarizado en una variedad de aplicaciones, entonces estas podrán acceder a las entidades compartidas de forma simultánea. Su principal desventaja es la sensibilidad del sistema de ventanas, ya que este componente puede verse afectado por las latencias de red entre los sitios participantes y el servidor. Estos efectos pueden ser mitigados mediante la introducción de cachés en los sitios participantes, lo cual implica un costo computacional adicional y un almacenamiento elevado [Lukosh, 2003].

3. **Arquitecturas híbridas:** estas arquitecturas tienen al menos un componente central y una comunicación directa entre los sitios de participantes. Es la combinación de las arquitecturas de componentes centralizados y de las que tienen comunicación directa. Algunos tipos de estas arquitecturas son el estado centralizado y la coordinación centralizada.

   - **Estado centralizado:** es una combinación de los estados centralizado y replicado. Los componentes de presentación se ubican en los sitios participantes, los cuáles están conectados entre sí y con el núcleo funcional colocado en el sitio servidor (ver figura 2.4a).
   - **Coordinación centralizada:** se integra por un componente de coordinación ubicado en el servidor y un núcleo de la aplicación instalado en cada sitio participante. Todos los componentes están conectados entre sí (ver figura 2.4b).

4. **Arquitecturas asimétricas:** este tipo de arquitecturas se genera en base a la combinacion de la arquitecturas centralizada, descentralizada e híbrida. Las arquitecturas asimétricas no tienen componentes centrales, sino que están distribuidos entre los sitios participantes de forma no simétrica, i.e. cada participante puede o no tener una réplica de algún componente. Una arquitectura asimétrica está integrada por varios sitios, los cuales se agrupan en pares. Así, cada par tiene características específicas de alguna de las arquitecturas de distribución (centralizada, replicada e híbrida) y dispone de sus propias entidades compartidas. Algunos tipos de estas arquitecturas son: la aplicación asimétrica, el estado asimétrico y la coordinación asimétrica.

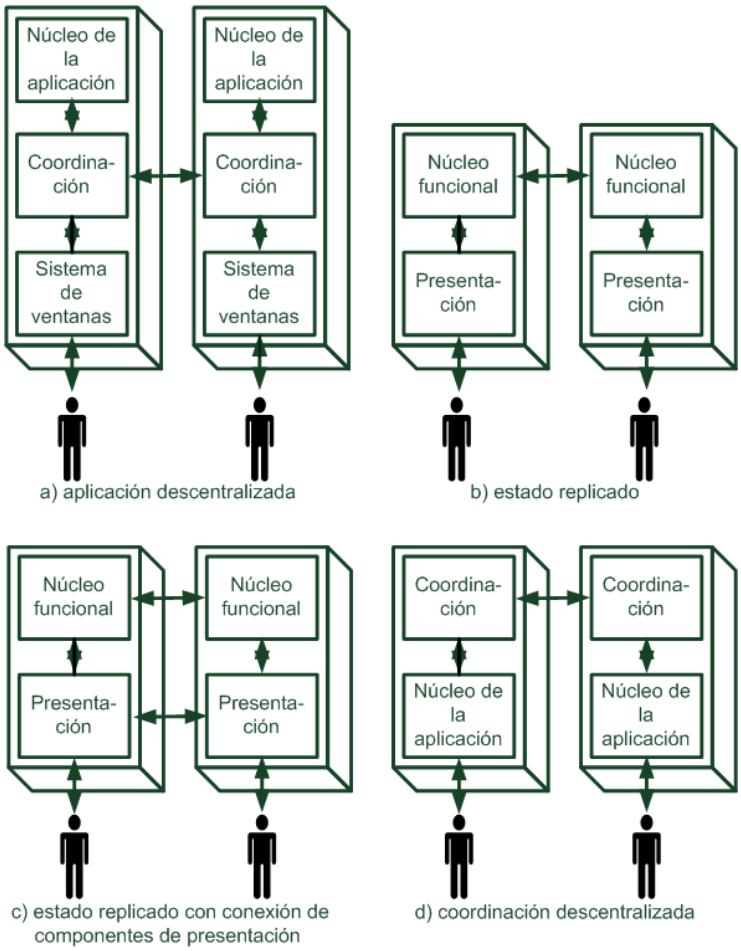

Figura 2.3: Arquitecturas con comunicación directa

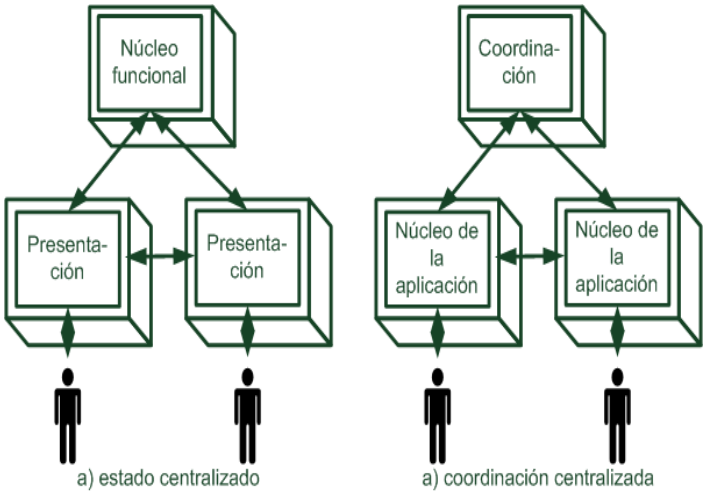

Figura 2.4: Arquitecturas híbridas

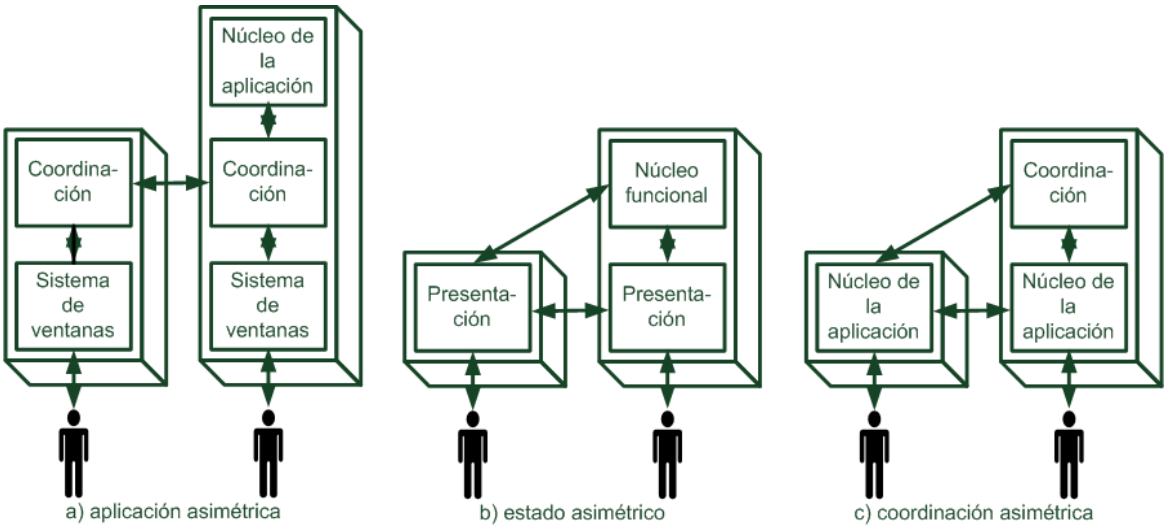

Figura 2.5: Arquitecturas asimétricas

- **Aplicación asimétrica**: un sitio participante puede tener todos los componentes (el núcleo de la aplicación, la coordinación y el sistema de ventanas), mientras que otro sitio puede contener únicamente el sistema de ventanas y la coordinación, siendo esta última el enlace entre ambos sitios (ver figura 2.5a).
- **Estado asimétrico**: algunos colaboradores pueden acceder localmente a las entidades compartidas, en tanto que los demás sitios participantes pueden interactuar con el sitio que contenga dichas entidades, ya que actuá como sitio de almacenamiento (ver figura 2.5b).
- **Coordinación asimétrica**: uno de los sitios participantes contiene el componente de la coordinación y del núcleo de la aplicación, mientras que el otro sitio sólo dispone del núcleo de la aplicación (ver figura 2.5c).

5. **Múltiples servidores:** estas arquitecturas utilizan más de un servidor i.e., existen componentes centrales y distribuidos en varios sitios (ver figura 2.6).

   - **Estado de distribución jerárquica**: se representa como un árbol binario en el cual la raíz y sus nodos hijos son núcleos funcionales y así consecutivamente hasta llegar al final de la rama, la cual está compuesta por un componente de presentación. Esta forma de organización permite extender el número de servidores y sitios participantes. Además, existe un control jerárquico de los diferentes componentes que forman la red, de manera que es posible definir grupos de colaboración. La principal ventaja de esta arquitectura es que los usuarios pueden acceder rápidamente al servidor de una red local. Un servidor dispone de más redes subordinadas mientras

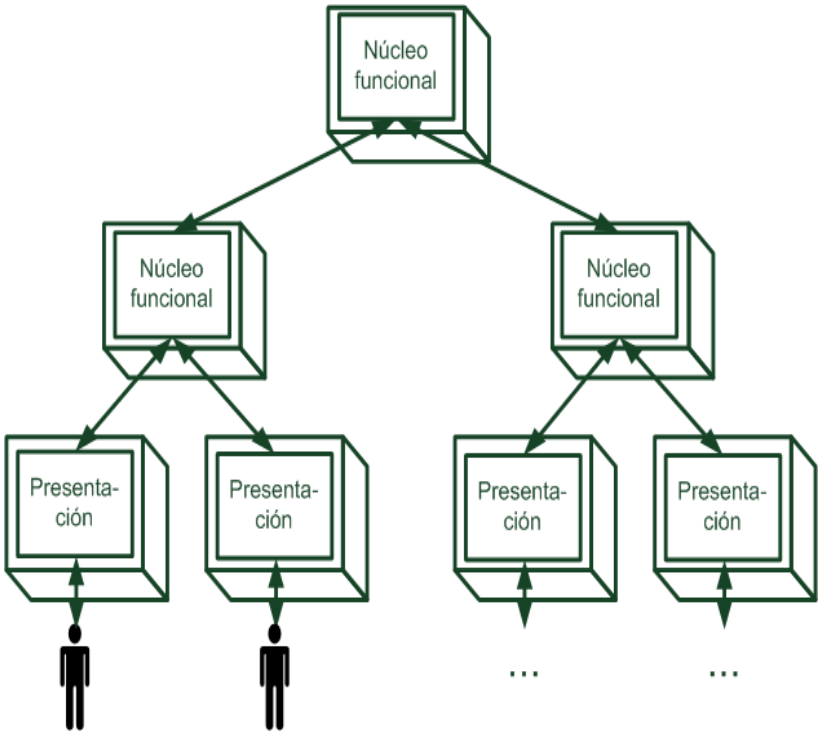

Figura 2.6: Arquitectura jerárquica de múltiples servidores

más alto esté en la jerarquía. Además tipo de arquitectura mejora la eficiencia en la distribución de las entidades compartidas y puede soportar otros tipos de arquitecturas con múltiples servidores. Su principal desventaja es que los algoritmos tienden a ser más complejos ya que se requiere mantener la consistencia.

2.2. Infraestructuras de trabajo colaborativo

El concepto de flexibilidad o adaptabilidad se refiere a que el desarrollador puede elegir qué tipo de arquitectura va a regir la distribución de cada una de las entidades compartidas del sistema colaborativo. Dicho proceso se debe realizar de forma automática conforme a ciertos factores, los cuales son mencionados a continuación.

En base a la clasificación precedente se puede decir que una arquitectura de distribución no puede ser apropiada para todas las circunstancias. Cada día más grupos requieren entornos de trabajo adaptables para el desarrollo de sus proyectos. Este requerimiento implica que el grupo pueda elegir un tipo de arquitectura de distribución, en tiempo de ejecución, en el transcurso de una sesión. Para lograr este objetivo el desarrollador debe considerar varios factores del entorno, tales como: las capacidades de los dispositivos receptores y servidores, la topología de la interconexión, el ancho de banda y la latencia de red, además de la variabilidad de cualquiera de estos factores

en el transcurso del tiempo [Phillips, 1999].

La mayoría de las infraestructuras soportan un solo tipo de arquitectura de distribución. En consecuencia, las entidades compartidas y sus componentes de tratamiento solo pueden distribuirse de una sola forma durante una sesión colaborativa. Algunos ejemplos de este tipo de infraestructuras son: Rendezvous [Hill et al., 1993], Habanero [NCSA], GroupKit [Roseman and Greenberg, 1996], Suite [Dewan and Choudhary, 1992], Notification Server [Patterson et al., 1996] y Dolphin [Streitz et al., 1994].

En la sección 2.2.1. se mencionan algunas de las infraestructuras flexibles que se enfocan en diferentes servicios, como: la consistencia de datos, los estilos de trabajo y el control de la concurrencia. En la sección 2.2.2. se explican algunas de las infraestructuras flexibles que dan soporte a diferentes esquemas de distribución de entidades.

2.2.1. Infraestructuras flexibles de distribución

Las infraestructuras enfocadas en la provisión de servicios flexibles de distribución ofrecen diversos esquemas, como el esquema adaptable o el esquema estático. A continuación, se describen las dos infraestructuras más relevantes:

GEN (GroupEnvironment): es un prototipo experimental que actualmente ya no es mantenido [O'Grady, 1996]. GEN disponía de un conjunto de herramientas para el desarrollo de aplicaciones colaborativas, utilizando arquitecturas centralizadas o replicadas. Sin embargo, esta infraestructura solo se enfocaba en redes de área local.

DreamObjects: soporta una variedad de arquitecturas de distribución, como el esquema de distribución adaptable y el esquema de distribución estático [Lukosch, 2003]:

- **Esquema de distribución adaptable**: este esquema cambia el conjunto de datos titulares (i.e., la representación local de las entidades compartidas) dependiendo de cómo se accedan. En base a las características del sitio, el conjunto de datos titulares puede ser modificado por un evaluador dinámico. DreamObjects ofrece dos evaluadores de este tipo:

  - **Evaluador orientado a usuario**: es adecuado cuando las entidades de la aplicación pueden ser divididas en piezas lógicas, e.g., un archivo de texto está estructurado en párrafos. Este evaluador adapta la distribución de las entidades compartidas en base al estilo de trabajo de cada colaborador. Justamente las entidades son distribuidas en los sitios que tienen accesos más intensivos.
  - **Evaluador orientado a red**: este evaluador trata de optimizar las entidades compartidas con respecto a la estructura de red del grupo colaborativo. Este esquema reduce el tráfico en la red, decrementa el tiempo de

respuesta y no requiere de un tipo especial de topología. Cuando un sitio participante inicia una sesión, el evaluador decide si este sitio se convierte en un titular de datos.

- **Esquema de distribución estático**: este esquema tiene predefinidos los conjuntos de datos titulares por el desarrollador antes de iniciar una sesión. La principal desventaja reside en la imposibilidad de cambiar los datos titulares durante el transcurso de dicha sesión. Este esquema es controlado por evaluadores que se implementan sobre las arquitecturas replicada y asimétrica**.**

  Un evaluador estático puede decidir si un sitio participante convierte una entidad particular cuando forme parte de una sesión o cuando sea creada. Adicionalmente, el evaluador estático determina qué sitio ejecuta o lee una llamada a un método y saca una balanza de la carga entre los diferentes datos de los sitios participantes, e.g., si un colaborador desarrolla una aplicación que es utilizada en una red de trabajo predefinida, entonces él puede definir su propio evaluador y distribuir una entidad compartida únicamente en los sitios que tienen una buena conexión de red. DreamObjects define dos tipos de evaluadores estáticos para controlar este esquema:

  - El evaluador predefinido para el **esquema de distribución asimétrico** es adecuado cuando el desarrollador quiere permitir a los usuarios de una sesión colaborativa introducir datos de forma local.
  - El **esquema de distribución replicado** es conveniente cuando cada sitio participante accede y modifica frecuentemente las entidades. En este caso, el esquema de distribución replicado decrementa el tiempo de respuesta en la interfaz de usuario.

  En conclusión, el esquema de distribución estático permite que el conjunto de datos titulares cambie solo cuando un sitio se une a la sesión en curso o bien cuando sale de esta.

  DreamObjects no puede ser implantado en un entorno de área amplia ni ofrece soporte para derechos de acceso.

2.2.2. Otras infraestructuras flexibles

Algunos servicios que toman en cuenta algunas infraestructuras para proveer flexibilidad son: la consistencia de datos, los estilos de trabajo y el control de concurrencia. A continuación se citan tres de estas infraestructuras.

Prospero: es una implementación abierta, la cual es usada para construir marcos flexibles de trabajo colaborativo. Al sistema Prospero originalmente se le intentó incorpar una distribución flexible, pero esta meta se eliminó más tarde a favor de proveer un nuevo mecanismo de mantenimiento flexible de la consistencia [Dourish, 1996].

DÁgora: está basada en el modelo de grupos de objetos iguales (*peer*). Los colaboradores pueden cambiar su estilo de trabajo durante una sesión, la cual se adapta a estos cambios de forma automática [Simao et al., 1997].

Clock: es una plataforma colaborativa, basada en componentes, que ofrece servicios de control de concurrencia. Además, el desarrollador de aplicaciones puede elegir entre una arquitectura de distribución centralizada o replicada. Las aplicaciones colaborativas están integradas por componentes que son especificados a través del propio lenguaje de programación de la infraestructura, llamado *functional Clock* [Graham et al., 1996b].

2.3. Movilidad de datos y código en la Web

Un aspecto importante en ambientes de trabajo cooperativo en la Web es la movilidad de las entidades compartidas y de los componentes de tratamiento entre los sitios que forman parte de dicho ambiente. Una arquitectura de distribución transmite dichas entidades y componentes al sitio participante **dónde** serán almacenados para su uso o ejecución según corresponda. La arquitectura también establece **cuándo** el usuario recibe retroalimentación de sus mismas acciones y de las acciones de otros participantes. En base a este enfoque se puede realizar dos tipos de clasificaciones: clasificación de entidades de datos y clasificación de componentes de tratamiento [Ramduny and Dix, 1997].

Clasificación de entidades de datos

Las entidades de datos pueden clasificarse en base al lugar donde son almacenadas y utilizadas (ver Figura 2.7). Los lugares de almacenamiento y de uso pueden ser referenciados como locales o remotos:

- el *caching* (copia temporal de datos) es utilizado localmente, ya que los datos originales están almacenados en otro sitio. Este esquema genera una rápida retroalimentación;
- la **replicación** se refiere a que el almacenamiento y el uso de las entidades de datos se realiza de forma local, lo que origina un tiempo de respuesta más rápido;
- el **acceso distante**, mediante llamadas de procedimientos remotos (RPC), significa que los datos se utilizan y almacenan remotamente, así que la información generada se transmite a los demás sitios participantes.

Uso / Almacenamiento	Local	Remoto
Local	Réplica	----------
Remoto	Caching	Acceso distante RPC

Figura 2.7: Uso y almacenamiento de entidades de datos

Ejecución / Almacenamiento	Local	Remoto
Local	Helpers	----------
Remoto	Java applets	Scripts CGI

Figura 2.8: Ejecución y almacenamiento de componentes de tratamiento

Clasificación de componentes de tratamiento

Los componentes de tratamiento se clasifican en base al lugar donde son almacenados y ejecutados (ver Figura 2.8):

- los *helpers* se almacenan y se ejecutan localmente,
- los *applets* se almacenan remotamente, pero se ejecutan localmente,
- los *scripts* CGI se almacenan y se ejecutan remotamente.

Es esencial decidir dónde se realiza la ejecución de los componentes de tratamiento con el fin de proveer una rápida retroalimentación. El costo de la distribución, hacia los sitios participantes, de cada cambio realizado en las entidades de datos compartidos depende del lugar donde estén almacenadas dichas entidades.

La Web provee un mecanismo llamado migración, que se refiere a la propiedad del servidor de almacenar código ejecutable (componente de tratamiento), el cual será transmitido al cliente para ser ejecutado localmente.

Capítulo 3

Lógica difusa y árboles de decisiones

En este capítulo, se da una síntesis de los orígenes de la lógica y de los conjuntos difusos, así como su desarrollo a partir de la lógica y de los conjuntos clásicos *(crisp)*. Particularmente se muestra su representación gráfica, las operaciones que soportan y las relaciones existentes en los conjuntos difusos. Además se ofrece una descripción formal (sintaxis, semántica e inferencia) de los sistemas basados en reglas. En particular, se presentan los sistemas de tipo Mamdani y Sugeno (dos de los más conocidos) los cuales sirven de base para entender mejor cómo se debe desarrollar un sistema difuso. Finalmente, se describen los árboles de decisión, los cuales permiten modelar y construir las reglas de inferencia que formarán la base de conocimiento del sistema difuso en cuestión. Algunas de las reglas que sirven para la creación de dichos árboles son la de división, parada y poda.

3.1. Lógica difusa

La lógica difusa fue concebida por Lofti A. Zaded, en respuesta a su inconformidad con los conjuntos clásicos *(crisp sets)* que sólo permiten dos opciones: la pertenencia o la no pertenencia de un elemento a un conjunto. Zadeh presentó a la lógica difusa como una forma de procesar información que permite pertenencias parciales a conjuntos que, en contraposición a los clásicos, nombró conjuntos difusos *(fuzzy sets)* [Zadeh, 1965].

Zadeh mencionó: "la lógica difusa trata de copiar la forma en que los humanos toman decisiones". Aunque se trabaja con información imprecisa esta lógica es en cierto modo muy precisa. Zadeh señaló que la gente no siempre requiere información numérica precisa del medio que lo rodea para desarrollar tareas de control altamente adaptables, e.g., conducir un automóvil o caminar por una acera sin chocar con los postes o con otras personas. Si los controladores convencionales, se pudieran programar para aceptar entradas con ruido e imprecisas, entonces dichos controladores

podrían trabajar de una manera más eficiente y quizás podrían ser implementadas más fácilmente.

En inteligencia artificial la lógica difusa, también conocida como lógica borrosa, se utiliza para la resolución de una variedad de problemas, relacionados principalmente con el control de procesos industriales complejos, la toma de decisiones, la resolución y la comprensión de datos, etc. Los sistemas de lógica difusa están presentes en la tecnología cotidiana, e.g., cámaras digitales, sistemas de aire acondicionado y brazos robóticos. Los sistemas basados en lógica difusa imitan la forma en que los humanos toman decisiones, pero tienen la ventaja de ser mucho más rápidos, robustos y tolerantes a imprecisiones y ruidos en los datos de entrada.

En la lógica tradicional o aristotélica, las computadoras manipulan valores duales i.e., que pertenecen o no pertenecen a un determinado conjunto, e.g., verdadero o falso, claro u oscuro, si o no. Por el contrario, en la lógica difusa, un valor real pertenece en un cierto porcentaje a cada uno de los conjuntos definidos (conjuntos > 2), e.g., frio-tibio-caliente o bajo-medio-alto.

Cabe destacar los excelentes resultados que brinda un sistema de control basado en lógica difusa: ofrece salidas de una forma veloz y precisa, disminuyendo así las transiciones entre los estados fundamentales del entorno físico que está siendo controlado, e.g., si el aire acondicionado se encendiera al llegar a la temperatura de $30°$ y la temperatura actual oscilara entre los $29° - 30°$, el sistema de aire acondicionado estaría encendiéndose y apagándose continuamente, con el gasto energético respectivo. Si dicho sistema estuviera regulado por lógica difusa, la temperatura de $30°$ no sería ningún umbral, ya que el sistema de control aprendería a mantener una temperatura estable sin continuos apagados y encendidos.

La lógica difusa es una metodología que proporciona una manera simple y elegante de obtener una conclusión a partir de información de entrada vaga, ambigua, imprecisa, con ruido o incompleta. No puede solucionar todos los problemas pero ayuda a modelar problemas difíciles.

3.1.1. Teoría de conjuntos clásicos (*crisp sets*)

Antes de presentar la teoría de conjuntos difusos, se describen brevemente los conjuntos clásicos, ya que éstos son la base para el desarrollo de los primeros. Algunos conceptos básicos se listan a continuación:

- **Conjunto clásico o *crisp set***: es una agrupación, clase o colección de objetos, denominados elementos del conjunto, que pueden compartir ciertas características.

- **Conjunto universal**: consiste de todos los elementos de interés para una aplicación particular.

- **Conjunto vacío**: carece de elementos.

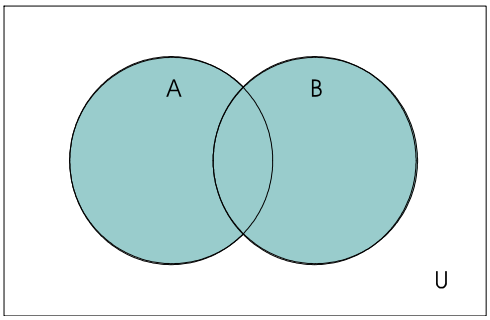

Figura 3.1: Unión de dos conjuntos clásicos

Operaciones de conjuntos

Las operaciones que sirven para manipular los conjuntos clásicos son la unión, la intersección, la diferencia y el complemento. A continuación se muestran gráficamente las operaciones de los conjuntos clásicos y su equivalencia difusa en grados de membresia [Ibrahim, 2004].

Unión: la unión de los conjuntos A y B está definido por:

$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

La función de membresia del conjunto A es $\mu_A(x)$ y del conjunto B es $\mu_B(x)$, entonces la membresia del conjunto resultante es $\mu_{A \cup B}(x)$ [Ibrahim, 2004]. Las representaciones gráficas clásica y difusa de la unión, se muestran respectivamente en las figuras 3.1 y 3.2.

Intersección: la intersección del conjunto A y B está definido por:

$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

La función de membresia del conjunto A es $\mu_A(x)$ y del conjunto B es $\mu_B(x)$, entonces la membresia del conjunto resultante es $\mu_{A \cap B}(x)$ [Ibrahim, 2004]. Las representaciones gráficas clásica y difusa de la intersección, se muestran respectivamente en las figuras 3.3 y 3.4.

Diferencia: la diferencia de los conjuntos A y B es un conjunto que consta de todos los elementos que pertenecen a A, pero no pertenecen a B.

$$A - B = A - A \cap B$$
$$= \{x \mid x \in A \text{ and } x \notin B\}$$
$$B - A = B - B \cap A$$
$$= \{x \mid x \in B \text{ and } x \notin A\}$$

En la figura 3.5 se muestra la representación gráfica clásica de la diferencia. Cabe mencionar que no existe la equivalencia de la diferencia en términos de función de membresía.

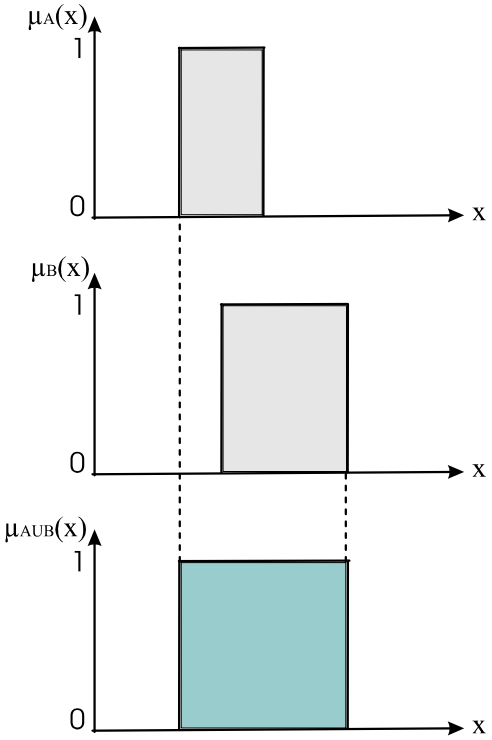

Figura 3.2: Unión en términos de función de membresía

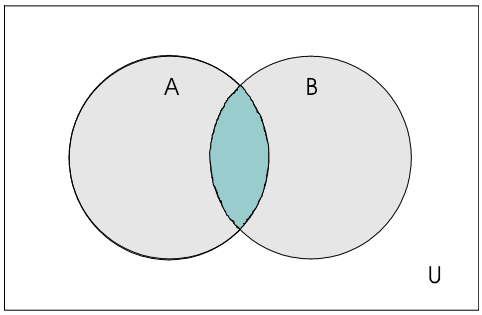

Figura 3.3: Intersección de dos conjuntos clásicos

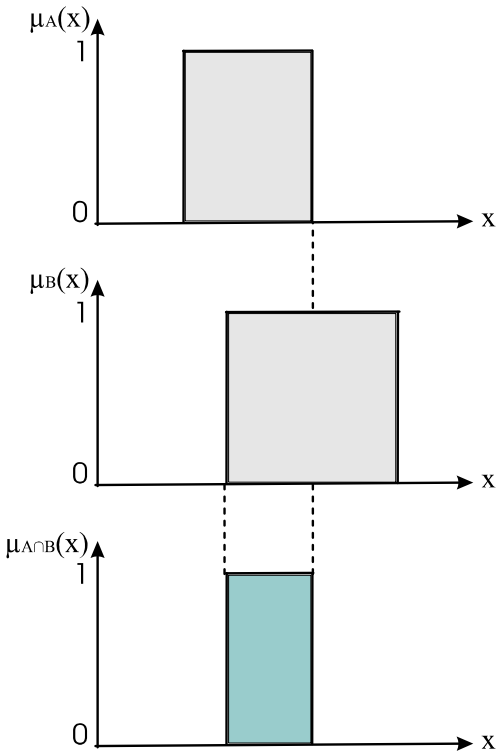

Figura 3.4: Intersección en términos de función de membresía

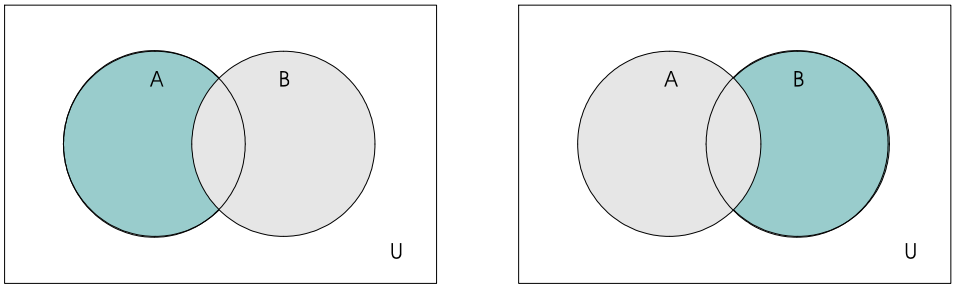

Figura 3.5: Diferencia de dos conjuntos clásicos

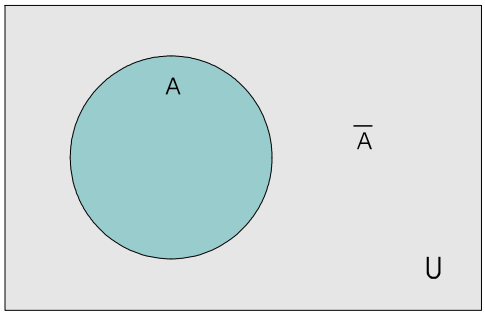

Figura 3.6: Complemento clásico del conjunto A

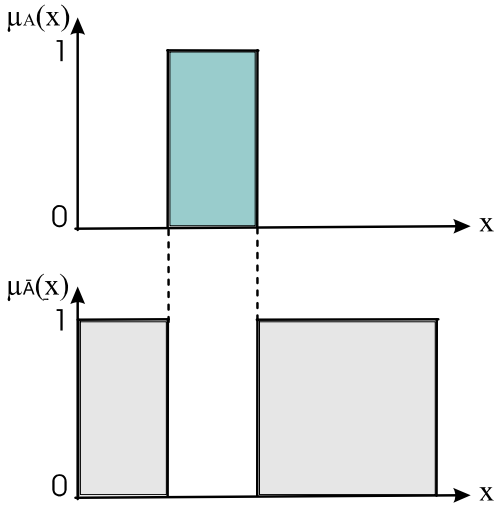

Figura 3.7: Complemento en términos de función de membresía

Complemento: el complemento del conjunto A, denotado por $\bar{A}$, contiene todos los elementos del universo U que no pertenecen a A.

$\bar{A}$ = U - A

$\bar{A}$ ={x | x ∈ U and x ∉ A}

La función de membresía del conjunto A es $\mu_A(x)$, entonces la membresía del conjunto resultante es $\mu_{\bar{A}}(x)$ [Ibrahim, 2004]. Las representaciones gráficas clásica y difusa del complemento se muestran respectivamente en las figuras 3.6 y 3.7.

A continuación se muestran las tablas de verdad de los operadores lógicos clásicos (AND, OR y NOT).

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

Tabla de verdad AND

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

Tabla de verdad OR

A	NOT A
0	1
1	0

Tabla de verdad NOT

3.1.2. Teoría de los conjuntos difusos

La lógica de los conjuntos difusos [Zadeh, 1965] trabaja con conjuntos que no tienen límites perfectamente definidos, i.e., la transición entre la pertenencia y no pertenencia de una variable a un conjunto es gradual. Esta lógica se caracteriza por las funciones de pertenencia que dan flexibilidad a la modelación, mediante el uso de expresiones lingüísticas tales como: frio-tibio-caliente, bajo-medio-alto, etc. Los conjuntos difusos surgieron de la necesidad de solucionar problemas complejos e imprecisos para los cuales la matemática y la lógica tradicionales no son suficientes. La lógica difusa ofrece un lenguaje que permite trasladar sentencias sofisticadas del lenguaje natural a un formalismo matemático. Aunque algunos conceptos pueden ser mejor definidos con palabras, los conjuntos difusos ayudan a construir mejores modelos de la realidad.

Representación gráfica de los conjuntos difusos

Los conjuntos difusos están formados por las siguientes partes [Zadeh, 1965] [Chahuara, 2005]:

- **Universo de discurso**: contiene todos los elementos que pueden ser tomados bajo consideración para asignar valores a las variables del sistema difuso. Los conjuntos difusos toman sus elementos del universo de discurso, el cual está denotado por la letra U.
- **Función de membresía**: todos los elementos del universo de discurso U son miembros, en cierto grado, de algún conjunto difuso. La función de membresía es la curva que define el grado con el que cada elemento está incluido en un conjunto difuso. Para la definición de las funciones de membresía se utilizan formas estándar como la función triangular, trapezoidal, tipo S, exponencial, singleton o tipo Pi. La representación gráfica de las funciones de membresía triangular y trapezoidal se muestran en la figura 3.8. Sus correspondientes expresiones matemáticas se muestran a continuación [Ibrahim, 2004].

1. Función de membresía triangular:

$$\mu(x) = a(b-x)/(b-c); \qquad b < x \leq c$$
$$= a(d-x)/(d-c); \qquad c < x \leq d$$

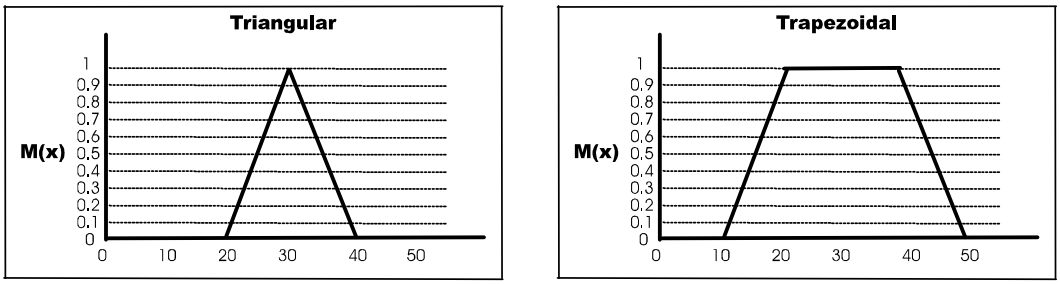

Figura 3.8: Funciones de membresía triangular y trapezoidal

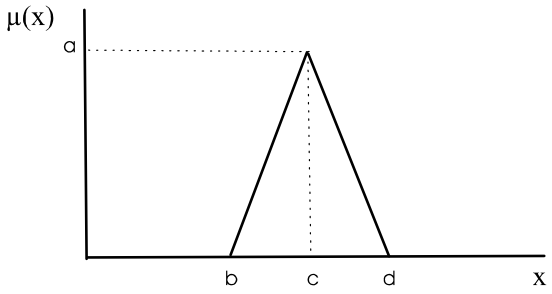

Figura 3.9: Función de membresía triangular

$= 0;$ $\quad otherwise$

En la figura 3.9 se muestra la correspondencia entre las variables de la función de membresía triangular y su representación gráfica.

2. Función de membresía trapezoidal:

$$\mu(x) = a(b-x)/(b-c); \quad b < x \leq c$$
$$= a; \quad c < x \leq d$$
$$= a(e-x)/(e-d); \quad d < x \leq e$$
$$= 0; \quad otherwise$$

En la figura 3.10 se representa la correspondencia gráfica de las variables de la función de membresía trapezoidal.

- **Variables lingüísticas**: son los elementos fundamentales de cualquier sistema de lógica difusa. Las variables lingüísticas combinan múltiples categorías subjetivas que describen el mismo concepto, e.g. para el caso de la variable `altura` existirán las categorías: `bajo, mediano, alto y muy alto,` las cuales son llamadas "términos lingüísticos" porque representan los posibles valores de una variable lingüística.

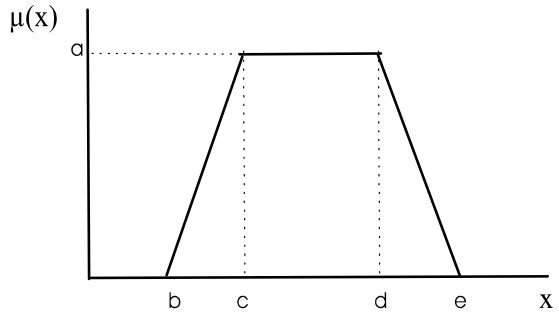

Figura 3.10: Función de membresía trapezoidal

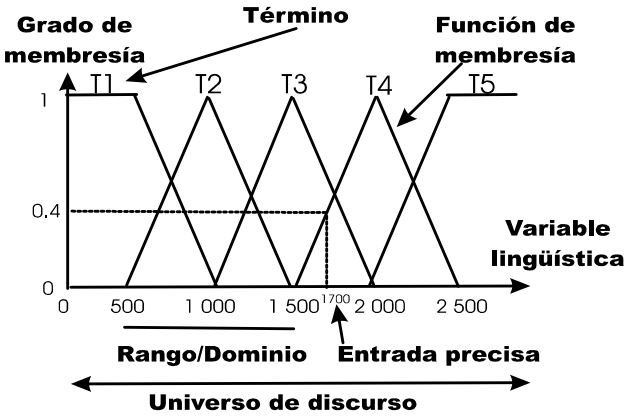

Figura 3.11: Conjuntos difusos

- **Grado de membresía**: es el grado con el cual una entrada bien definida es compatible con una función de membresía. Puede tomar valores entre 0 y 1 (ver figura 3.11).
- **Término**: es una categoría subjetiva de una variable lingüística, en consecuencia es el nombre descriptivo utilizado para identificar una función de membresía. Tal como las variables algebraicas toman valores numéricos, las variables lingüísticas toman términos lingüísticos.
- **Entrada precisa**: son los diferentes valores discretos de una variable del sistema, e.g., las alturas de un grupo de personas son 1.50 m, 1.60 m y 1.70 m.
- **Rango/dominio**: es el intervalo sobre el cual se define una función de membresía, e.g., una función de membresía `alto` podría tener un dominio de 1.6 a 1.9 m y un rango de 0.3 m.

La lógica difusa proporciona definiciones de conjuntos que tienen límites difusos *(fuzzy)* en lugar de los límites nítidos *(crisp)* de la lógica de Aristóteles. Estos con-

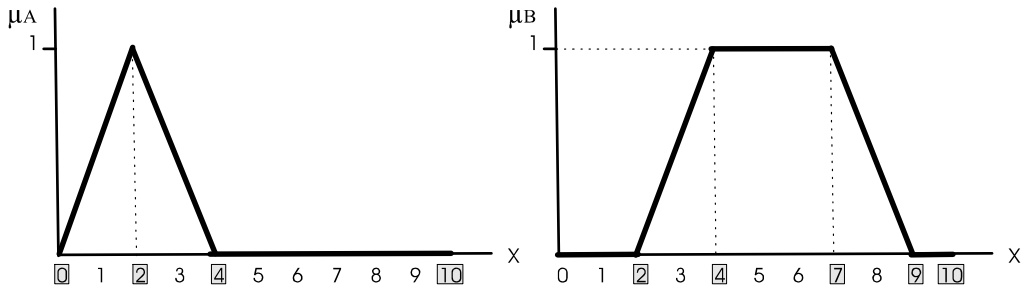

Figura 3.12: Conjuntos difusos triangular A y trapezoidal B

juntos pueden traslaparse de modo que, para un valor específico de entrada, uno o más conjuntos asociados a una etiqueta lingüística pueden simultáneamente ser verdaderos hasta un cierto grado. Si una **entrada precisa** tiene una membresía alta en un determinado conjunto y dicha entrada se mueve hacia un conjunto adyacente, entonces el primer conjunto se vuelve progresivamente menos verdadero, mientras que el segundo conjunto llega a ser progresivamente más verdadero.

Los conjuntos difusos brindan un medio para utilizar expresiones lingüísticas, como "la temperatura es tibia", en reglas que pueden ser evaluadas con un alto grado de precisión numérica y repetidamente. Esto contradice la percepción errónea tan común de que la lógica difusa produce resultados aproximados. De hecho un conjunto específico de condiciones de entrada siempre produce el mismo resultado.

Operaciones entre conjuntos difusos

Como se explicó anteriormente las operaciones básicas de unión, intersección y complemento de los conjuntos clásicos son aplicables a los conjuntos difusos [Zadeh, 1965] [Ibrahim, 2004]. En la figura 3.12 se muestran dos conjuntos difusos mediante las funciones de membresía triangular y trapezoidal. En el eje de las x, algunos números están encerrados en cuadros, con el fin de representar e identificar el área que abarca la función de membresía.

- **Unión**: la unión de dos conjuntos difusos A y B (ver figura 3.13) es un conjunto difuso cuya función de membresia está dada por:

  $\mu_{A\cup B}(x) = \text{máx}[\mu_A(x), \mu_B(x)]$

- **Intersección**: la intersección de dos conjuntos difusos A y B (ver figura 3.14) es otro conjunto difuso, el cual tiene una función de membresia dada por:

  $\mu_{A\cap B}(x) = \text{mín}[\mu_A(x), \mu_B(x)]$

- **Complemento**: el complemento absoluto (ver figura 3.15) de un conjunto difuso es denotado por $\bar{\text{A}}$, cuya función de membresía está definida por:

  $\mu_{\bar{A}}(x) = 1 - \mu_{\bar{A}}(x)$ para todo $x \in X$

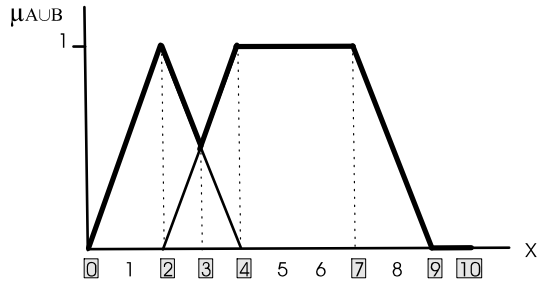

Figura 3.13: Unión de los conjuntos A y B

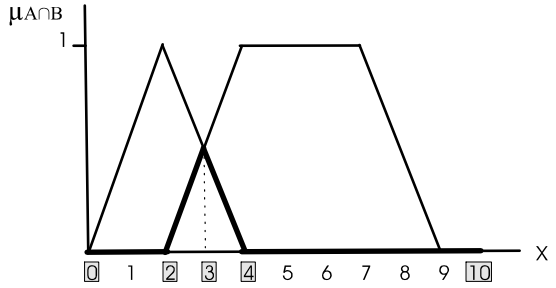

Figura 3.14: Intersección de los conjuntos A y B

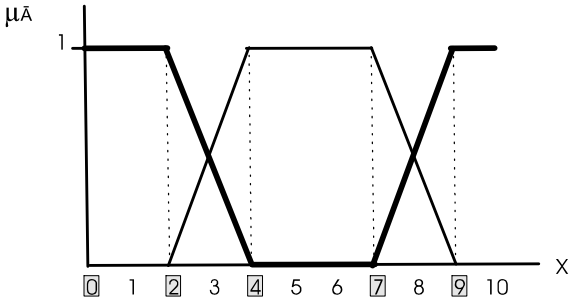

Figura 3.15: Complemento del conjunto A

Estas operaciones cumplen las propiedades de asociatividad, conmutatividad y distributividad así como las leyes de Morgan.

Las funciones que definen la unión y la intersección de conjuntos difusos pueden generalizarse a condición de cumplir determinadas restricciones. Las funciones que cumplen estas condiciones se conocen como Conorma Triangular (o T-Conorma) y Norma Triangular (o T-Norma). Los principales operadores que cubren las condiciones para ser T-Conormas son el operador máximo $[\mu_{A\cup B}(x) = \mu_A(x)\,\text{máx}\,\mu_B(x)]$ y la suma algebraica $[\mu_{A\cup B}(x) = (\mu_A(x) + \mu_B(x)) - (\mu_A(x) - \mu_B(x))]$. Los principales operadores que cubren las condiciones para ser T-Normas son el operador mínimo $[\mu_{A\cap B}(x) = \mu_A(x)\,\text{mín}\,\mu_B(x)]$ y el producto algebraico $[\mu_{A\cap B}(x) = \mu_A(x) \cdot \mu_B(x)]$. En la mayoría de las aplicaciones de la lógica difusa a la ingeniería se usan como T-Conorma el operador máximo y como T-Norma los operadores mínimo o producto.

Se puede considerar que la lógica clásica es un caso límite de lógica difusa. En consecuencia, las tablas de verdad de los operadores lógicos clásicos y de los operadores lógicos difusos deben coincidir de la siguiente forma:

- la **intersección (AND)** de dos conjuntos difusos A y B se modela mediante la familia de operadores T-Normas, siendo el mínimo y el producto algebraico los dos casos de T-Normas más sencillos y utilizados.
- la **unión (OR)** de dos conjuntos difusos A y B se modela mediante la familia de operadores T-Conormas, cuyos representantes más comunes son el máximo y la suma algebraica.
- la familia de operadores para la **negación (NOT)** del conjunto A es la complementariedad aditiva.

Las tablas siguientes conciernen a los operadores de los conjuntos difusos.

A	B	T-norma(A,B)
0	0	0
0	1	0
1	0	0
1	1	1

Tabla de verdad T-Norma

A	B	T-conor(A,B)
0	0	0
0	1	1
1	0	1
1	1	1

Tabla de verdad T-Conorma

A	Complemento
0	1
1	0

Complemento

Relaciones difusas

Una relación difusa [Zadeh, 1971] [Ibrahim, 2004] representa el grado de presencia o ausencia de asociación, interacción o interconexión entre elementos de dos o más conjuntos difusos, e.g., "a es mayor que b". Sean U y V dos universos de discurso, la relación difusa *R(U,V)* es un conjunto difuso en el espacio producto $U \cdot V$ que se caracteriza por la función de membresía $\mu_R(x, y)$ donde x pertenece a U e y pertenece a V, i.e., $R(U, V) = \{((x, y), \mu_R(x, y))|(x, y) \in U \cdot V\}$. En el caso de las relaciones difusas, $\mu_R(x, y) \in [0, 1]$ y en el caso de las relaciones clásicas, $\mu_R(x, y) = 0$ o 1.

Las relaciones difusas son, en si mismas, un conjunto difuso en el espacio producto. Las operaciones entre conjuntos (unión, intersección y complemento) y los operadores (máximo, suma algebraica, mínimo y producto algebraico) se pueden aplicar a dichas relaciones difusas, e.g., $R(x, y)$ y $S(x, y)$ en el mismo espacio producto $U \cdot V$. La intersección y la unión entre R y S, que son composiciones entre las dos relaciones, se definen como:

$\mu_{R \cap S}(x, y) = \mu_R(x, y) \bigotimes \mu_s(x, y)$

$\mu_{R \cup S}(x, y) = \mu_R(x, y) \bigoplus \mu_s(x, y)$

Donde $\bigotimes$ es cualquier T-Norma y $\bigoplus$ es cualquier T-Conorma.

Si se toman las relaciones difusas R y S, que pertenecen a diferentes espacios de producto $R(U, V)$ y $S(V, W)$, e.g., "x es mayor que y" e "y es cercano a z", su composición difusa se define de forma análoga a la composición clásica: la relación difusa R tiene asociada una función característica $\mu_R(x, y)$ que toma valores en el intervalo $[0, 1]$; en tanto que la relación difusa S también tiene asociada una función característica $\mu_s(y, z)$ que toma valores en el intervalo $[0, 1]$. Entonces, la **composición difusa** entre R y S, i.e., $R \circ S$, cuando R y S pertenecen a universos discretos de discurso, se define como una relación difusa en $U \cdot W$ cuya función de membresía está dada por:

$\mu_{R \circ S}(x, z) = \sup[\mu_R(x, y) \bigotimes \mu_S(y, z)]$

$y \in V$

donde el operador *sup* es el máximo y el operador $\bigotimes$ puede ser cualquier T-Norma. En función de la T-Norma elegida, se obtienen distintas composiciones como la *máx-mín* y la *máx-product*, etc.

- La composición *máx-mín* de las relaciones difusas $R(U, V)$ y $S(V, W)$ es una relación difusa $R \circ S$ en $U \cdot W$ definida por la función de membresía:

  $\mu_{R \circ S}(x, z) = máx - mín[\mu_R(x, y), \mu_S(y, z)]$

  $y \in V$

  donde $(x, z) \in U \cdot W$

- La composición *máx-product* de las relaciones difusas $R(U, V)$ y $S(V, W)$ es una relación difusa $R \circ S$ en $U \cdot W$ definida por la función de membresía:

  $\mu_{R \circ S}(x, z) = máx[\mu_R(x, y) \cdot \mu_S(y, z)]$

  $y \in V$

  donde $(x, z) \in U \cdot W$

3.1.3. Descripción formal de un sistema difuso

Los sistemas difusos [Chahuara, 2005] están basados en reglas, las cuales se expresan como implicaciones lógicas (SI-ENTONCES). Dichas implicaciones pertenecen a la lógica matemática y reflejan la relación que guarda un hecho derivado de otro.

Se puede decir que una lógica consta de:

1. Un **sistema formal**, que describe lo que está sucediendo en un momento determinado y que está formado por:

   - La **sintaxis del lenguaje**, que explica como construir oraciones.
   - La **semántica del lenguaje**, por medio de la cual cada oración expresa algo relacionado con el mundo.

2. Una **teoría de demostración**, que agrupa un conjunto de reglas para deducir las implicaciones de un conjunto de oraciones y que especifica los pasos de razonamiento confiables.

Para entender la relación entre la lógica proposicional y la lógica difusa [Chahuara, 2005] se deben tomar en cuenta la sintaxis, la semántica y la inferencia de cada una de ellas:

Sintaxis:

- **Sintaxis de la lógica proposicional**: está integrada por las constantes lógicas V (verdadero) y F (falso), los operadores lógicos conjunción ($\wedge$), disyunción ($\vee$) y negación ($\sim$), la equivalencia ($\longleftrightarrow$), la implicación ($\rightarrow$) y los símbolos de las proposiciones como p y q.

- **Sintaxis de la lógica difusa**: en lugar de constantes lógicas V y F se utilizan grados de verdad, grados de membresia o grados de pertenencia a los conjuntos difusos que varían entre 0 y 1. Los operadores lógicos son la negación difusa, las T-Normas y las T-Conormas, en tanto que los símbolos de las proposiciones son iguales a los de la lógica proposicional.

Semántica:

En las lógicas proposicional y difusa, la semántica se define mediante la especificación de la interpretación de los símbolos de proposición, de las constantes y de los conectores lógicos.

- **Semántica de la lógica proposicional**: para cualquier situación solo existe una de dos posibles respuestas (falso o verdadero).

- **Semántica de la lógica difusa**: se basa en la suposición de que las situaciones tienen cierto grado de verdad, así que pueden existir más de dos posibles respuestas.

Inferencia:

La inferencia lógica es un proceso mediante el cual se implanta la relación de implicación que existe entre oraciones, i.e., si se cuenta con determinadas premisas y una posible conclusión, entonces se puede deducir mediante aspectos lógicos si dicha conclusión es verdadera.

Según la Wikipedia, la inferencia lógica es un proceso mental por el cual se extraen conclusiones a partir de premisas más o menos explícitas. Este proceso resulta ya sea del sentido común, ya sea de los silogismos o bien de la aplicación de reglas muy detalladas y de cálculos de la inferencia estadística, e.g., cuando se escucha decir: "después de un debate cansado, el orador se dirigió a su silla" se puede inferir que el orador se sentó en ella. Como se puede observar la inferencia es un mecanismo por el cual se puede percibir un significado más o menos hipotético en base a pistas suficientemente claras.

Los procedimientos para asegurar la confiabilidad de una inferencia son las tablas de verdad y las reglas difusas [Chahuara, 2005]. Las tablas de verdad se pueden emplear en clases enteras de inferencias, pues existen ciertos patrones que se presentan una y otra vez. En lo que respecta a las reglas de inferencia, se utilizan cuando ya se tiene o se conoce el patrón adecuado, lo cual servirá para definir la regla de inferencia correspondiente. Algunos modelos de reglas de inferencia son: Modus Ponens, Modus Tolens, la resolución y la O-introducción.

- **Inferencia de la lógica proposicional**: cuando se aplica esta lógica a la ingeniería se utiliza el tipo de regla de inferencia Modus Ponens, el cual satisface la "ley de causa y consecuencia". Modus Ponens se utiliza para probar la corrección o el valor de verdad de las proposiciones lógicas, e.g.,

  Premisa 1 (hecho) -> "x es *alto*"

  Premisa 2 (conocimiento) -> "Si x es *alto*, entonces y es *bajo*"

  Conclusión -> "y es *bajo*"

- **Inferencia de la lógica difusa**: como se mencionó anteriormente, se utilizan las reglas difusas cuando ya se tiene definido algún patrón correspondiente a cierta situación. En la inferencia difusa se utiliza el modelo de reglas de inferencia Modus Ponens Generalizado, el cual es muy similar al de la lógica proposicional, e.g.,

  Premisa 1 (hecho) -> "a es *alto**"

  Premisa 2 (conocimiento) -> "Si a es *alto**, entonces b es *bajo**"

  Consecuencia -> "b es *bajo**"

Para determinar las principales diferencias entre el Modus Ponens y el Modus Ponens Generalizado se observa que `alto` no es necesariamente igual a `alto*` y de forma similar `bajo` no es igual a `bajo*`. En lógica proposicional una regla será disparada sólo

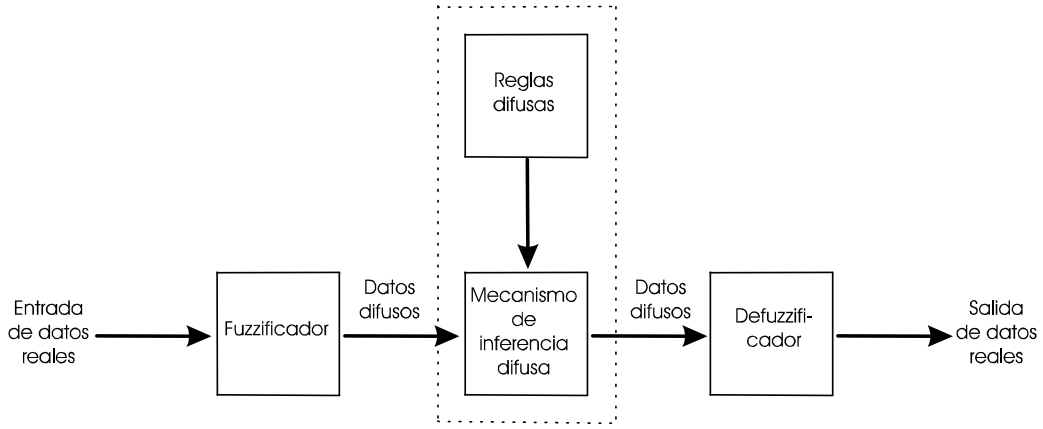

Figura 3.16: Procesamiento general de un sistema difuso Mamdani

si la primera premisa es exactamente igual al antecedente de esa regla, i.e., el resultado de tal activación es el consecuente real de la regla. Por el contrario, en lógica difusa una regla es activada mientras no haya un grado cero de similitud entre la premisa y el antecedente de la regla, i.e., el resultado de la activación es un consecuente que tiene un grado de similitud diferente de cero con respecto al consecuente de la regla.

3.1.4. Tipos de sistemas lógicos difusos

Un sistema lógico difuso [Chahuara, 2005] es un mapeo no lineal de un vector de datos de entrada con una salida escalar (mapeo de números con números). La lógica difusa y los conjuntos difusos definen las especificaciones para realizar un mapeo no lineal, ya que se pueden manejar datos numéricos y conceptos lingüísticos de forma simultánea. Los tipos más populares de sistemas son:

- Sistemas difusos de tipo Mamdani
- Sistemas difusos de tipo Sugeno

Sistemas de tipo Mamdani

En un sistema difuso de tipo Mamdani [Chahuara, 2005] tanto el antecedente como el consecuente de las reglas están dadas por expresiones lingüísticas, e.g., "si la entrada es baja entonces la salida es alta". En la figura 3.16 se muestra su estructura general.

Fuzzificador: la entrada de un sistema de tipo Mamdani es un valor numérico proveniente de un sensor, un archivo, etc. Es necesario convertir este valor numérico en un valor difuso para que pueda ser interpretado por el mecanismo de inferencia.

Mecanismo de inferencia difusa: una vez que el fuzzificador arrojó los valores difusos, el mecanismo de inferencia los procesa para obtener un grado de membresía a un determinado subconjunto difuso. Posteriormente, utilizando dicho grado de membresía, se realiza la búsqueda de la regla más adecuada en la base de reglas difusas para obtener la salida difusa más apropiada.

Base de reglas difusas (base de conocimiento): es la manera que tiene el sistema difuso de guardar el conocimiento lingüístico, que le permite resolver el problema para el cual está diseñado. Como se mencionó anteriormente en la descripción formal, estas reglas son del tipo SI-ENTONCES, el cual está formado por dos partes: el antecedente y el consecuente.

SI la entrada es baja **ENTONCES** la salida es alta
Antecedente Consecuente

Defuzzificador: la salida que genera el mecanismo de inferencia es una salida difusa, la cual no puede ser interpretada por una entidad externa (e.g., un controlador) que solo manipula información numérica. Para lograr que la salida del sistema difuso pueda ser interpretada, es necesario convertirla al tipo de salida que entienda la entidad externa. Hay varias opciones para realizar dicha conversión:

- Centro de Gravedad $y = \frac{\sum_i b_i \int \mu(i)}{\sum_i \int \mu(i)}$ y
- Centros Promediados $y = \frac{\sum_i b_i \mu_{premisa(i)}}{\sum_i \mu_{premisa}(i)}$.

Sistemas de tipo Sugeno

Otra alternativa de procesamiento en los sistemas difusos fue propuesta por Sugeno (ver figura 3.17). Este modelo difiere del de Mamdani en que no cuenta con un defuzzificador [Chahuara, 2005].

Fuzzificador: realiza la misma función que el fuzzificador de Mamdani.

Mecanismo de inferencia difusa: idem que la descripción del mecanismo de inferencia difusa de Mamdani.

Base de reglas difusas: las reglas de la base de conocimiento de un sistema Sugeno son diferentes de las reglas de un sistema Mamdani, pues el consecuente de estas reglas ya no es una etiqueta lingüística, sino una función de la entrada del sistema en un momento dado, como se ilustra a continuación:

SI la entrada es baja **ENTONCES** la salida = función (entrada)
Antecedente Consecuente

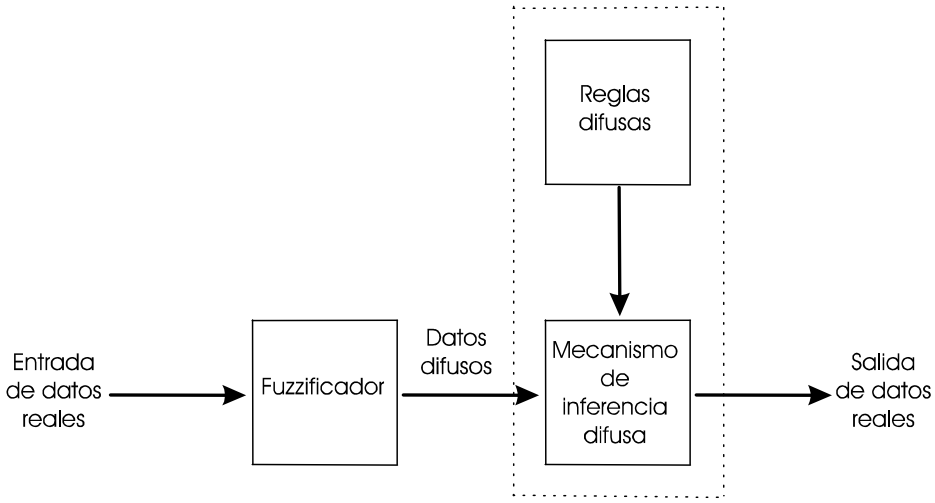

Figura 3.17: Procesamiento general de un sistema difuso Sugeno

El resultado arrojado (consecuente) por la regla difusa activa es un valor numérico real que puede ser utilizado por la entidad externa adecuada. Por tal motivo ya no se requiere de una etapa de defuzzificación como en el caso de los sistemas Mamdani.

3.2. Árboles de decisión

Los árboles de decisión son una herramienta eficiente para listar, contar y analizar los posibles resultados en una sucesión de eventos. Uno de los principios de la generación de árboles es la **regla del producto**, cuya definición informal sería [Micha, 1998]: "cuando un evento ocurre en varias etapas, se tiene que multiplicar el número de maneras de realizar cada una de las etapas para encontrar el total de formas en que el evento sucede".

Suponga que un proceso consiste de dos pasos: si hay n_1 maneras de hacer el primer paso y n_2 maneras de realizar el segundo paso, entonces hay $n_1 \cdot n_2$ maneras de efectuar el proceso completo, e.g., si $n_1 = 2$ y $n_2 = 2$, entonces el árbol tendrá 4 posibilidades ya que $2 \cdot 2 = 4$. Este principio se puede extender a los procesos que consisten de más de dos pasos y más de dos maneras. Suponga que hay k pasos, si hay n_1 maneras de hacer el primer paso, n_2 maneras de realizar el segundo paso, ..., y n_k maneras de efectuar el último paso, entonces hay $n_1 \cdot n_2 \cdot ... \cdot n_k$ maneras de llevar a cabo el proceso completo.

Un árbol de decisión [Berzal, 2002] puede utilizarse para clasificar un ejemplo concreto, comenzando en su raíz y siguiendo el camino determinado por las respuestas a las preguntas de los nodos internos, hasta llegar a una hoja del árbol.

Según la Wikipedia, algunas definiciones de árboles de decisión son:

Definición 1: es un modelo de predicción utilizado en el ámbito de la inteligencia

artificial. Dada una base de datos, se generan diagramas de contrucciones lógicas muy similares a los sistemas de predicción basados en reglas. Dichos diagramas sirven para representar y categorizar una serie de condiciones que suceden de forma sucesiva para encontrar la solución de un problema determinado.

Definición 2: es una herramienta para ayudar a realizar elecciones adecuadas entre muchas posibilidades. Su estructura permite seleccionar una y otra vez diferentes opciones para explorar las diferentes alternativas posibles de decisión.

Definición 3: es la representación gráfica de distintas opciones posibles, cada una de las cuales a su vez afecta a otras. Un árbol de decisión se utiliza en el cálculo de probabilidades para tomar la decisión que parece más segura o adecuada en base a las entradas recibidas.

Un árbol de decisión tiene entradas, las cuales pueden ser un objeto o una situación descrita por medio de un **conjunto de atributos**. A partir de dichas entradas se da una respuesta, la cual es una decisión tomada en base a las entradas. Los valores que pueden tomar dichas entradas y salidas pueden ser discretos o continuos. Cuando se hace uso de valores discretos se le denomina **clasificación** y cuando se utilizan valores continuos se le llama **regresión**.

Un árbol de decisión lleva a cabo una prueba a medida que este se recorre hacia las hojas para alcanzar una decisión. El árbol de decisión suele contener nodos internos, nodos de probabilidad, nodos hojas y ramas. Un nodo interno contiene una prueba sobre un valor de alguna de las propiedades. Un nodo de probabilidad indica que debe ocurrir un evento aleatorio de acuerdo a la naturaleza del problema. Un nodo hoja representa el valor que devolverá el árbol de decisión y finalmente las ramas brindan los posibles caminos a seguir de acuerdo a la decisión tomada.

Los árboles de decisión [Berzal, 2002] son útiles siempre que los ejemplos, a partir de los que se desea aprender, se puedan representar mediante un conjunto prefijado de atributos y valores discretos o continuos. Cuando los ejemplos son variables, este tipo de árbol no es adecuado.

3.2.1. Procedimiento para construir árboles de decisión

Los árboles de decisión (clasificación o regresión) [Berzal, 2002] son un método de aprendizaje supervisado, el cual se utiliza para resolver problemas de clasificación de todo tipo. El conocimiento obtenido durante el proceso de aprendizaje se representa mediante un árbol, donde cada nodo interno contiene una pregunta acerca de un atributo particular y cada hoja hace referencia a una decisión.

En algunas ocasiones, no es posible construir modelos que permitan clasificar todos los posibles casos, especialmente cuando se ha de tratar con muchos atributos o valores desconocidos o cuando algunos atributos deben modelarse en función de otros [Berzal, 2002]. A veces sólo se pueden descubrir modelos aproximados o parciales, los cuales contemplan algunas características de las distintas clases, pero no abarcan todas las posibles clases ni todos los casos particulares de una clase determinada. Para abordar este tipo de problema, se puede considerar la clase como un atributo

más o bien se puede utilizar reglas de asociación para dividir el conjunto de casos de entrenamiento [Berzal, 2002].

Los árboles de decisión se construyen recursivamente [Berzal, 2002] siguiendo una estrategia descendente, desde los conceptos generales hasta llegar a los particulares. Inicialmente se deben reunir los datos que se utilizarán como base del conjunto de entrenamiento. Posteriormente se descartan los atributos irrelevantes utilizando algún método de elección de características. Por último se construye recursivamente el árbol de decisión en base al principio de divide y venceras.

Las reglas que deben seguirse para la construcción de árboles de decisiones son [Berzal, 2002]: reglas de división, reglas de parada y reglas de poda.

Reglas de división: cualquier pregunta que divida el conjunto de casos de entrenamiento en al menos dos subconjuntos no vacíos conducirá a la construcción de un árbol de decisión. Los criterios de división generalmente se basan en las medidas de impureza de un nodo. Este término se refiere al grado en que un nodo incluye casos de distintas clases del problema de clasificación a resolver. Un nodo puro es aquel al que sólo corresponden casos pertenecientes a una de las clases del problema. Un ejemplo de regla de división es la de divide y vencerás.

Reglas de parada: estas reglas sirven para detener la construcción de un árbol de decisiones. Cuando esto sucede se construye una hoja a la que se le puede asignar una distribución de probabilidad o la clase más común de las existentes en los casos recogidos. A las reglas que tratan de predecir si merece o no la pena seguir construyendo el árbol por la rama actual se les llama reglas de pre-poda. Existe otro tipo de reglas llamadas post-poda que simplifican el árbol de decisión una vez que este ha sido construido por completo. Existen varios criterios por los que se debe detener la construcción del árbol, e.g. cuando se ha llegado a un nodo puro (que contiene ejemplos de una única clase) o cuando se establece una cota de profundidad (que evita la construcción de árboles grandes y/o complejos).

Reglas de poda: la presencia de ruido ocasiona la generación de árboles de decisión muy complejos. Para evitar este problema se utilizan técnicas de poda, según las cuales aquellas ramas con menor capacidad de predicción suelen ser podadas una vez que el árbol ha sido construido completamente.

3.2.2. Transformación de árboles en reglas de inferencia

Una cualidad de los árboles de decisión [Berzal, 2002] es la posibilidad de generar un conjunto de reglas de inferencia del tipo SI-ENTONCES, en base a las ramificaciones del árbol. Es necesario tomar en cuenta que si el problema de clasificación es complejo, entonces el árbol resultante será de igual índole, por lo que ni siquiera podándolo se asegura la comprensión completa del modelo de clasificación.

El procedimiento para obtener el conjunto de reglas [Berzal, 2002] consiste en recorrer cada una de las ramas del árbol iniciando en la raíz y terminando en los nodos hojas. Las reglas resultantes son una conjunción de las literales relativas a los valores de los atributos, los cuales se sitúan en los nodos internos del árbol. El consecuente de las reglas resultantes es la decisión a la que hace referencia la hoja del árbol. Dichas reglas serán mutuamente excluyentes. Si las reglas generadas contienen condiciones irrelevantes en su antecedente, entonces podrán generalizarse (eliminando las condiciones redundantes) sin que disminuya la precisión del clasificador asociado. En este caso las reglas pueden dejar de ser mutuamente excluyentes.

En la figura 3.18 se muestra un diagrama de decisión con las siguientes partes:

- En la parte izquierda de la página está localizado un recuadro que representa el inicio del árbol. Sobre este recuadro se encuentra la decisión que se requiere tomar.
- A partir de este recuadro existe una línea dirigida hacia la derecha por cada posible solución. Sobre cada línea se encuentra una solución.
- Al final de cada línea se estima un resultado. Un círculo pequeño significa que el resultado es incierto, mientras que un recuadro representa una nueva decisión que requiere ser tomada. El resultado incierto y la nueva decisión se localizan respectivamente arriba del círculo y del recuadro.
- A partir del recuadro de una nueva decisión, existen líneas que representen las opciones que se pueden seleccionar. Similarmente, a partir de cada círculo existen líneas que representan las posibles consecuencias. Nuevamente se encontrará una inscripción sobre cada línea que indica su significado.

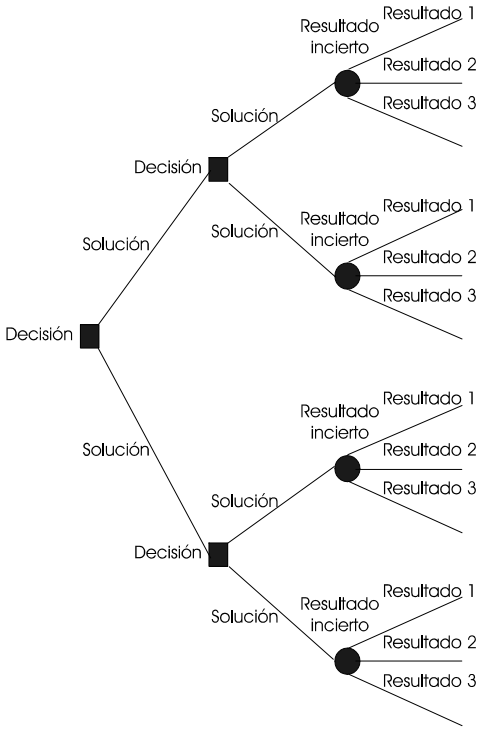

Figura 3.18: Diagrama de decisión

Capítulo 4

Arquitecturas y mecanismos de comunicación

Las arquitecturas de comunicación se clasifican en arquitecturas cliente-servidor y *peer to peer*, siendo estás últimas una evolución de las primeras. Estas arquitecturas funcionan mediante protocolos que proveen servicios específicos, como es el caso del protocolo de transferencia de hipertexto (HTTP). Dado que este protocolo ha sido utilizado en el desarrollo de la presente tesis, se explican sus principales características y su relación con los modelos OSI y TCP/IP. Asimismo, se describen algunos servidores Web como Apache, AOLServer, Jetty y Thttpd. Otros aspectos importantes de Internet que se tratan en este capítulo son algunos de sus servicios más importantes, la denominación de sitios, el estándar MIME y su relación con dichos servicios. Finalmente, se describe la metodología CGI, así como la evolución de los lenguajes utilizados en el desarrollo de aplicaciones Web.

4.1. Categorías principales de arquitecturas

Una arquitectura de comunicación es un sistema en el que se conectan entre sí varios sitios independientes para compartir datos, recursos de cómputo y periféricos, tales como discos duros e impresoras. El funcionamiento de una arquitectura de comunicación requiere del *software* que posibilita el envío y la recepción de datos, así como la administración de la propia red.

Las arquitecturas de comunicación pueden clasificarse en dos categorías: 1) arquitecturas cliente-servidor y 2) arquitecturas *peer to peer*, las cuales se describen a continuación.

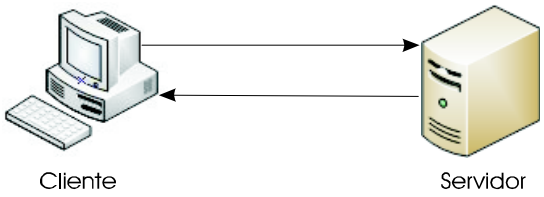

Figura 4.1: Arquitectura cliente-servidor

4.1.1. Arquitectura cliente-servidor

La arquitectura cliente-servidor se puede definir como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. Los principales elementos que conforman este tipo de arquitectura son:

1. **Cliente**: es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor. A este proceso también se le conoce como *front-end.*

2. **Servidor**: es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por el servidor. A este proceso también se le conoce como *back-end.*

3. **Mensaje**: es el mecanismo utilizado para la petición de servicios y la entrega de resultados entre clientes y servidores.

En el ámbito de TCP/IP, la comunicación entre computadoras básicamente funciona bajo la arquitectura cliente-servidor. Mediante esta arquitectura se busca proveer de flexibilidad, interoperabilidad, escalibilidad y usabilidad a las comunicaciones. La comunicación entre cliente y servidor comienza con un mensaje enviado por el cliente, solicitando un determinado servicio al servidor, el cual procesa el mensaje y remite uno o varios mensajes con la respuesta pertinente.

Las computadoras pueden realizar diversas tareas pero, en base a sus características y capacidades de cómputo, ejecutan unas mejor que otras. Los clientes por lo general tienen menos capacidades que los servidores. Así, los clientes sirven para ejecutar la interfaz gráfica de usuario de una aplicación cliente-servidor, mientras que los servidores realizan la administración de datos, la gestión de recursos compartidos, etc. Esta arquitectura permite distribuir físicamente los procesos y los datos en forma eficiente, con el fin de reducir el tráfico en la red (ver figura 4.1).

Las arquitecturas cliente/servidor utilizan uno o varios servidores que, además de garantizar la seguridad de los archivos y directorios, están optimizados para dar una respuesta eficiente a las peticiones de los clientes. Conforme aumente el tamaño de la red, se requerirá que más servidores se especialicen en determinadas tareas. Así cada tarea será realizada de una forma más eficiente.

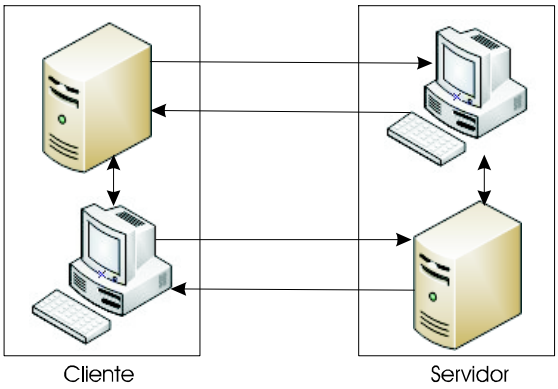

Figura 4.2: Arquitectura *peer to peer*

4.1.2. Arquitectura *peer to peer*

Las arquitecturas *peer to peer* [Millan, 2006] no son un concepto nuevo. De hecho son la evolución natural de la arquitectura cliente-servidor utilizada en Internet. Esta arquitectura se vuelve más conocida conforme pasa el tiempo gracias al avance de la tecnología en los siguientes factores: a) incremento en el ancho de banda, b) mayor número de computadoras conectadas a Internet, c) mayores capacidades de almacenamiento e d) incremento de fuentes servidoras de información y contenido.

Una arquitectura P2P no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan a la vez como clientes y servidores de los demás nodos de la red. Todos los nodos actuán de igual forma, aunque pueden tener diferentes configuraciones. Así cada nodo puede descargar/cargar archivos de/a otros nodos de la red (ver figura 4.2).

4.2. Protocolo de transferencia de hipertexto

La World Wide Web (WWW) es uno de los servicios más utilizados de Internet. Su éxito se debe en gran medida al protocolo HTTP y al lenguaje HTML. Dicho protocolo proporciona una implementación simple y sencilla de un sistema de comunicaciones que permite fácilmente el envío de cualquier tipo de archivo.

Es necesario distinguir dos conceptos fundamentales: servicio y protocolo. Internet se utilizó inicialmente de muchas maneras y con muchos propósitos. Por lo tanto, a cada una de estas formas de uso se le conoce como servicio de Internet, ya que es una forma estandarizada de utilización. Cada servicio requiere el uso de protocolos aceptados por clientes y proveedores de servicio.

Un protocolo de comunicación [Alonso, 2007] es un conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre computadoras conectadas a una red.

4.2.1. Modelos OSI y TCP/IP

Para establecer el proceso de comunicación entre dispositivos de cómputo es necesaria la existencia de normas establecidas por organismos internacionales que regulen dicho proceso. Las normas estándares más comunes [Raya y Raya, 2002] que permiten la intercomunicación de computadoras son:

- OSI (*Open System Interconnection* o Interconexión de Sistemas Abiertos).
- TCP/IP (*Transmission Control Protocol/Internet Protocol* o Protocolo de Control de Transmisión/Protocolo Internet).

Los modelos OSI y TCP/IP categorizan a un conjunto de protocolos de comunicación mediante niveles. Cada nivel soluciona una serie de problemas relacionados con la transmisión de datos y proporciona un servicio bien definido a los niveles superiores. Mientras más alto es el nivel, los protocolos serán más cercanos al usuario ya que los datos manejados son más abstractos. Mientras más bajo es el nivel, los protocolos correspondientes se encargarán de traducir los datos de forma que sean físicamente manipulables.

OSI es uno de los modelos de interconexión de sistemas abiertos más utilizados. En la figura 4.3 se muestra la estructura general del modelo OSI [Coulouris et al, 2001], el cual se divide en siete niveles, cada uno conformado por un grupo de protocolos [Alonso, 2007] [Raya y Raya, 2007]:

1. **Nivel físico**: se definen y reglamentan todas las características físicas, mecánicas y eléctricas que debe cumplir el sistema para poder operar. Se puede incluir en esta capa los siguientes protocolos: cable coaxial, par trenzado, fibra óptica, microondas y ondas de radio.

2. **Nivel de enlace de datos**: en esta capa, el protocolo físico adecuado es asignado a los datos. Además, se establece el tipo de red y la secuencia de paquetes utilizada. Algunos protocolos de esta capa son: Ethernet, Token Ring, FDDI y ATM.

3. **Nivel de red**: esta capa determina la forma en que los datos serán enviados al dispositivo receptor. Aquí se manejan los protocolos de enrutamiento y la administración de direcciones IP. Los protocolos correspondientes a esta capa son: ARP, IP, NetBEUI, IPX, ICMP y IGMP.

4. **Nivel de transporte**: esta capa mantiene el control de flujo de datos y proveé verificación de errores y recuperación de datos entre dispositivos. El control de flujo significa que la capa de transporte verifica si los datos provienen de más de una aplicación y si es así, los integra en un solo flujo dentro de la red física. Los protocolos pertenecientes a este nivel son: TCP, UDP y SPX.

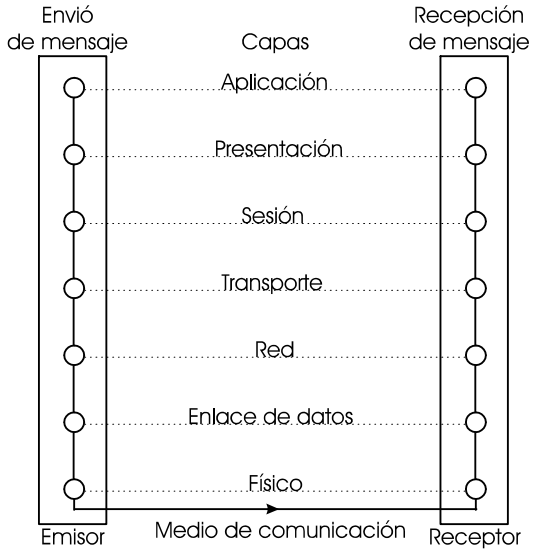

Figura 4.3: Capas del modelo OSI

5. **Nivel de sesión**: esta capa establece, mantiene y termina las comunicaciones que se establecen entre dispositivos. Los protocolos correspondientes a este nivel son: NetBIOS, RPC y SSL.

6. **Nivel de presentación**: esta capa tiene la misión de tomar los datos que han sido entregados por la capa de aplicación y convertirlos en un formato estándar que otras capas puedan entender. Un protocolo correspondiente a este nivel es: ASN.1.

7. **Nivel de aplicación**: esta es la capa que interactúa con el sistema operativo o aplicación cuando el usuario decide transferir archivos, leer mensajes o realizar otras actividades de red. Los protocolos pertenecientes a este nivel son: HTTP, FTP, DNS, SMTP, POP3, IMAP, SSH y NFS.

TCP/IP es un conjunto de protocolos de red que utiliza Internet para la transmisión de datos entre redes de computadoras. TCP/IP es una referencia a los dos protocolos más importantes que lo integran, el Protocolo de Control de Transmisión y el Protocolo de Internet. Dichos protocolos son dos de los primeros que fueron definidos, además de que son los más frecuentemente utilizados.

La búsqueda de una relación o delimitación entre ambos modelos es difícil. TCP/IP puede describirse [Alonso, 2007] por analogía con el modelo OSI, ya que ambos categorizan a los protocolos en niveles, aunque este último no corresponde exactamente con el primero. TCP/IP fue diseñado para solucionar un problema práctico de ingeniería, mientras que el modelo OSI tiene un enfoque teórico, aunque se debe tener presente que dicho enfoque sirvió como base para la evolución de las redes informáticas. En conclusión, se puede decir que el modelo OSI es más fácil de entender, pero el modelo

TCP/IP es el que realmente se utiliza. A continuación, se resumen los cinco niveles TCP/IP (ver figura 4.4) [Coulouris et al, 2001].

1. **Nivel físico**: describe las características físicas de la comunicación, como las convenciones sobre la naturaleza del medio utilizado para la comunicación y todo lo relativo a conectores, código de canales y modulación, potencias de señal, longitudes de onda, sincronización y distancias máximas soportadas. Algunos de los protocolos correspondientes a este nivel son el par trenzado, el cable coaxial, la fibra óptica y las ondas de radio.

2. **Nivel de enlace de datos**: especifica información detallada de la forma en que se envían físicamente los datos a través de la red. Define la forma en que se realiza la señalización eléctrica de los bits, mediante dispositivos de hardware que conectan directamente con una red, e.g., cable coaxial. Los protocolos correspondientes a este nivel son: Ethernet, Token Ring, FDDI, X.25, Frame Relay, RS-232 y v.35.

3. **Nivel de Internet**: empaqueta los datos en datagramas IP, que contienen información de las direcciones origen y destino. Dicha información es utilizada para reenviar los datagramas a *hosts* y redes. Además, realiza el enrutamiento de los datagramas IP. Algunos protocolos pertenecientes a este nivel son: IP, ICMP, ARP y RARP.

4. **Nivel de transporte**: permite administrar las sesiones de comunicación entre equipos host. Asimismo, define el nivel de servicio y el estado de la conexión utilizada al transportar datos. Los protocolos referentes a este nivel son: TCP, UDP y RTP.

5. **Nivel de aplicación**: define los protocolos de aplicación TCP/IP y la forma en que se conectan los programas de *host* a los servicios del nivel de transporte para utilizar la red. Algunos protocolos correspondientes a esta capa son: HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP y X Windows.

4.2.2. Funcionamiento del protocolo HTTP

El protocolo de transferencia de hipertexto (HTTP) es un protocolo orientado a conexión, ya que hace uso del protocolo de control de transporte (TCP) para establecer un canal de comunicación, entre el cliente y el servidor, por el cual HTTP realiza la transferencia de datos. Su funcionamiento se explica a continuación (ver figura 4.5):

- el cliente establece una conexión con el servidor en el puerto TCP correspondiente a HTTP;

- sobre dicha conexión se envía un comando HTTP de petición; y

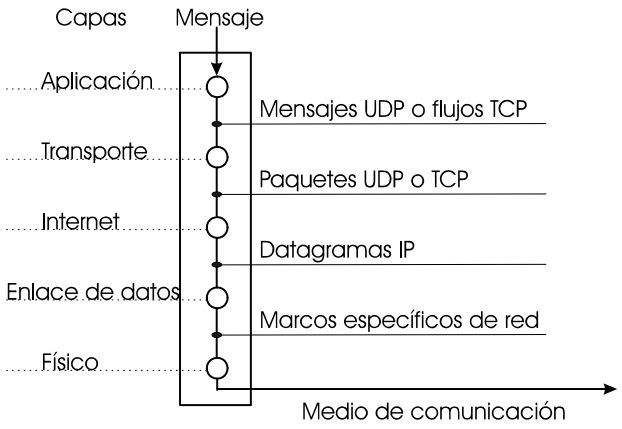

Figura 4.4: Capas del modelo TCP-IP

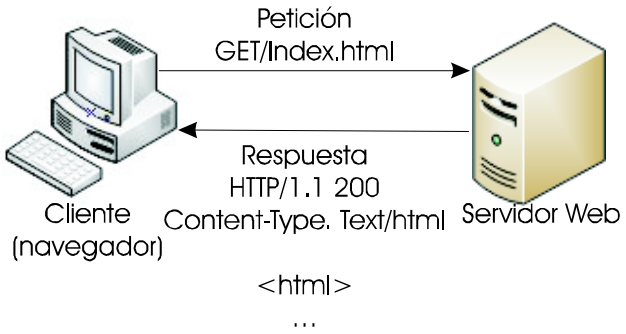

Figura 4.5: Funcionamiento del protocolo HTTP

- por esta misma conexión, el servidor transmite los datos o recursos solicitados al cliente.

HTTP utiliza el puerto 80 como predeterminado, aunque este número puede cambiar. Sin embargo, todos los puertos menores de 1024 están reservados. Así, se podrá establecer otro número de puerto siempre que se encuentre en el intervalo de 1024 a 65535 y que no esté siendo utilizado por algún otro programa.

4.2.3. Servidores Web

Un servidor Web es un programa que atiende y responde a las diversas peticiones que realizan los programas cliente (navegadores). El protocolo estándar utilizado por dicho servidor es HTTP, pero también se puede utilizar HTTPS que es la versión segura, cifrada y autenticada de HTTP. El puerto predeterminado para HTTPS es el 443.

El funcionamiento elemental de un servidor Web [Mateu, 2004] que sólo provee archivos estáticos consiste en:

1. esperar peticiones en el puerto TCP asignado (80 que es el predeterminado o algún otro dependiendo de la configuración),
2. recibir una petición,
3. buscar el recurso en la cadena de petición (URL),
4. enviar el recurso en la misma conexión por donde se recibió la petición, y
5. volver al punto 1.

En base a los pasos anteriores, se han diseñado y construido los servidores HTTP existentes. La diferencia entre ellos reside en el tipo de peticiones que atienden, ya que ahora no solamente sirven páginas estáticas sino también páginas dinámicas mediante *servlets,* CGI, etc.

- **Servicio de archivos estáticos**: todo servidor Web es capaz de servir los archivos estáticos que se encuentren en alguna parte concreta del disco duro. Por lo tanto, se debe poder especificar la ruta donde están contenidos dichos archivos (sistema de ficheros virtual). Un ejemplo se muestra en la siguiente tabla:

Directorio del disco	**Directorio Web**
/home/Apache/html	/
/home/Organización/docs	/docs
/home/usuario_x/manuales-2008	/manuales-2008

- **Contenido dinámico**: otra función importante de los servidores Web es la provisión de contenido dinámico. Actualmente, todos los servidores ofrecen soporte para *scripts* CGI, que es el método más antiguo y simple para generar contenido dinámico. Los *scripts* CGI pueden ser desarrollados en varios lenguajes como: C, PHP, JSP, ASP, etc. Cuando se elige un lenguaje se tiene que considerar si el servidor lo soporta, en caso contrario se tendrá que utilizar adicionalmente algún *software* externo como en el caso de los JSP.

4.2.4. Servidores Web de código libre

Existen varios servidores HTTP de código libre que ofrecen características y funcionalidades interesantes [Mateu, 2004]:

- **Apache**: la implementación de Apache se ha realizado de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. Es el servidor Web más implantado entre los distintos servidores que ofrecen servicios Web en Internet. Entre las características más significativas se destacan: es modular, tiene la capacidad de crear servidores virtuales y permite crear servidores seguros https y sitios privados.

- **AOLServer**: el servidor Web AOLServer fue desarrollado por América Online (AOL), el cual es uno de los proveedores más importantes de Internet en el mundo. AOLServer es un servidor Web multi-hilo con muchas facilidades de uso en entornos de gran escala y sitios Web dinámicos.

- **Roxen y Caudium**: Roxen es un servidor Web bajo la licencia GNU. Este servidor ofrece varios módulos a los usuarios mediante los cuales se pueden desarrollar sitios Web dinámicos, sin necesidad de utilizar otra herramienta más que el propio servidor. Sus características principales son:

  - es multiplataforma (Windows, Linux, Solaris y MAC OS/X);
  - dispone de una interfaz de administración vía la Web;
  - tiene un soporte gráfico integrado, el cual permite generar imágenes, gráficas, etc., por medio de algunas etiquetas de RXML (extensión de HTML de Roxen);
  - permite el acceso a bases de datos como PostgreSQL, Oracle, MySQL, etc.
  - tiene integrado el gestor de base de datos MySQL;
  - maneja algunos lenguajes como RXML, CGI, Java, Perl y PHP;
  - tiene soporte de criptografía;
  - ofrece una arquitectura modular que permite cargar y descargar extensiones del servidor cuando está en marcha.

Caudium es un proyecto basado en la versión del sevidor Roxen 1.3 mediante el cual se busca la compatibilidad con las versiones anteriores, ya que la versión 2.0 de Roxen no disponía de dicha compatibilidad. En la actualidad los desarrolladores tratan de mantener la compatibilidad entre las API de estos dos servidores.

- **Thttpd**: es un servidor extremadamente pequeño, muy rápido, portable y seguro. Dispone de las mismas prestaciones que otros servidores (Apache). Se usa muy poco como servidor de propósito general, sin embargo se emplea como auxiliar de Apache para servir contenido estático.

- **Jetty**: es un servidor Web escrito en Java que incluye un contenedor de *servlets*. Tiene un tamaño reducido y un alto rendimiento, por lo que es adecuado para desarrollar productos embebidos que necesiten de un servidor HTTP. Por lo general, este servidor no se suele encontrar funcionando por sí mismo, sino como servidor Web empotrado en varios productos:

  - está integrado en el proyecto JXTA como base para el transporte HTTP, así como en CDs de demostración en libros sobre Java, *servlets*, XML, etc.
  - se ejecuta en múltiples sistemas embebidos y PDAs.
  - está integrado en servidores de aplicaciones como JBoss y Jonas, así como en los productos Tívoli de IBM, MQ de Sonic y SESM de Cisco.

- **IIS (Internet Information Server)**: IIS [Beato y Franco, 1998] es en esencia un conjunto de servidores, destinado a Internet e Intranet, que está integrado por un servidor HTTP, FTP y Gopher. Entre sus características se pueden citar las siguientes:

  - se integra con los servicios proporcionados por el sistema operativo, así que se puede considerar como otro servicio NT;
  - tiene una herramienta gráfica de configuración, la cual permite la administración y la seguridad del servidor;
  - puede ser administrado remotamente mediante el uso de un navegador;
  - se integra con el sistema de seguridad de Windows NT; los usuarios pueden acceder a las páginas en función de los permisos definidos sobre estas;
  - permite realizar la publicación de páginas mediante herramientas Web (e.g., FrontPage);
  - permite utilizar mecanismos de seguridad avanzados como la autentificación cifrada de NT;
  - se integra fácilmente con el servidor SQL de Microsoft;
  - permite crear páginas activas y *scripts* mediante ASP (Active Server Pages o Páginas Activas de Servidor), CGI e ISAPI que añaden dinamismo a las páginas.

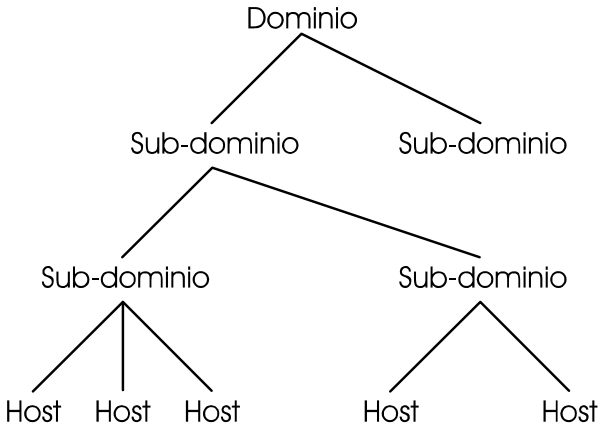

Figura 4.6: Estructura del dominio de una red

Los servidores IIS son constantemente atacados por virus, ya que estos aprovechan las características y vulnerabilidades detectadas en los servicios proporcionados por Windows NT.

4.2.5. Localización y acceso de recursos en Internet

Para identificar un equipo de cómputo en Internet, se utiliza el nombre del equipo y el nombre del dominio de la red. Este último es el nombre que se la da a la red para su identificación en Internet. La estructura del dominio es similar a un árbol (ver figura 4.6).

La identificación de un dominio está formado por varias partes separadas por un punto [Raya y Raya, 2002]. Cada parte de estas recibe el nombre de subdominio. Mientras el subdominio se encuentre más a la derecha tiene un carácter más general por lo que se le conoce como dominio. El nombre del dominio ha de empezar con el nombre del *host*, un punto, el nombre de la red local (subdominio) y así consecutivamente con las redes superiores. La identificación completa de un *host* en Internet sería: *host*.subdominio.subdominio.dominio e.g. cheverny.cs.cinvestav.mx. Para acceder a la cuenta de un usuario se tiene que referenciar el nombre del usuario, una arroba, el nombre del *host*, un punto, el subdominio y así consecutivamente.

Por otra parte, URL es el acrónimo de *Uniform Resource Locator* o Localizador Uniforme de Recursos, el cual permite localizar y acceder de forma sencilla a cualquier recurso de la red desde un navegador Web.

Por medio de la WWW, se pretende unificar el acceso a información de servicios que antes eran incompatibles entre sí. De esta forma, desde un mismo programa se puede tener acceso uniforme a todos los recursos y permitir que los documentos HTML incluyan enlaces a otras fuentes de información contenidas en servicios como FTP, correo electrónico, etc.

Las partes que integran un URL son: el nombre del servicio, dos puntos, dos

diagonales, el nombre del servidor, los subdominios, el dominio, dos puntos, el número del puerto TCP, una diagonal, la ruta de acceso al recurso, una diagonal, el nombre del recurso, e.g., http://cheverny.cs.cinvestav.mx:2008/cgi-bin/test-cgi.

- **Nombre del servicio**: es alguno de los que están disponibles en Internet, e.g., http, https, ftp, grupo de noticias, correo electrónico, telnet, gopher y wais.

- **Nombre del servidor**: es el nombre del servidor donde está almacenado el recurso.

- **Subdominios y dominio**: sirve para identificar a la organización, institución o país a la que pertenece la red, e.g., `.net (servicios de red)`, `.org (organización)`, `.edu (instituciones educativas)`, `. mx (méxico)`, `.com (comercial)`, `.gov (organismos gubernamentales)`, `.int (internacional)` y `.mil (organismos militares)`.

- **Número del puerto TCP**: es opcional, por lo regular no se pone siempre y cuando el puerto utilizado sea el predeterminado para el servicio en cuestión. Solo se utilizará cuando el servidor tenga un puerto distinto al predeterminado.

- **Ruta de acceso al recurso**: es el camino donde se encuentra el recurso. Para acceder a los subdirectorios se utiliza la barra /".

- **Nombre del recurso**: es el nombre del recurso y su extensión correspondiente. La extensión del recurso es importante, ya que mediante esta el servidor sabe el tipo de documento a servir e indica al cliente (e.g. navegador) el modo en que debe tratarse dicho recurso. Los tipos de los recursos se definen en el archivo MIME.

4.2.6. Tipos MIME

Los tipos *MIME* (*Multipurpose Internet Mail Extensions*, o Extensiones Multipropósito de Correo en Internet) es un estándar propuesto por los laboratorios Bell Communications para ampliar las posibilidades del correo electrónico, mediante la inserción de archivos (texto, imágenes, animaciones, audio y video) en un mensaje.

Desde entonces, el tipo MIME se utiliza para dar formato tanto a los documentos adjuntos en un mensaje como a los documentos transferidos a través del protocolo HTTP. Durante una transacción entre un servidor Web y un navegador, el servidor envía en primer lugar el tipo MIME del archivo al navegador para que sepa cómo tratarlo.

A continuación, se muestra una tabla en la que se listan algunos ejemplos de tipos MIME.

Tipos MIME	Tipo de archivo	Extensión asociada
text/html	Archivos HTML	htm, html
text/richtext	Archivos de texto enriquecido	rtx
text/plain	Archivos de texto sin formato	txt, text, conf, def, list, log, in
image/tiff	Imágenes tiff	tiff, tif
image/x-portable-anymap	Archivos Anymap PBM	pnm
image/x-rgb	Imágenes RGB	rgb
image/gif	Animación GIF	gif
audio/basic	Archivos de audio básicos	au, snd
audio/x-aiff	Archivos de audio AIFF	aif, aiff, aifc
audio/x-wav	Archivos de audio Wave	wav
video/mpeg	Video MPEG	mpeg, mpg, mpe, m1v, m2v
video/quicktime	Videos de QuickTime	qt, mov
video/x-sgi-movie	Videos de MoviePlayer	movie

La estructura que maneja el archivo `mime.types` es en dos columnas: la primera incluye el tipo `mime` principal, una diagonal y el subtipo `mime` y la segunda columna contiene las extensiones.

4.3. Scripts CGI

Un *script* es un fichero de texto que contiene un conjunto de sentencias del *shell* (if, case, for, etc), variables de ambiente y/o comandos, los cuales permiten implementar comandos de forma rápida, pero se requiere tener permiso de ejecución.

Todo proceso UNIX tiene una `entrada`, una `salida` y un `error estándar`. Estos son flujos de datos sobre los cuales pueden acceder determinadas funciones de los lenguajes de programación para la lectura y/o escritura de información.

4.3.1. CGI (interfaz común de puerta de enlace)

CGI se deriva de principios de programación fuertemente ligados a UNIX. Al igual que la mayoría de las entidades en Internet, un *script* CGI experimenta cambios periódicos. Se debe tener en cuenta que existen muchas extensiones y variaciones que pueden o no funcionar con algunos navegadores y servidores. CGI es un conjunto de variables y convenciones nombradas comúnmente para pasar información en ambos sentidos, entre el servidor y el cliente, utilizando la entrada, salida y error estándar [Rowe, 1996]. En otras palabras, un *script* CGI es una aplicación auxiliar interna que amplia las capacidades del servidor, ya que realiza tareas especificas de acuerdo a los requerimientos de los usuarios.

Un *script* CGI puede ser escrito prácticamente en cualquier lenguaje. Los criterios [Rowe, 1996] a tomar en cuenta para seleccionar el lenguaje son:

- el lenguaje debe estar soportado por el sistema operativo en el que corre el servidor;
- el programador debe estar suficientemente familiarizado con el lenguaje para trabajar de manera eficiente; y
- el lenguaje debe tener las facilidades necesarias para realizar las tareas que el programador necesita.

Algunos de los lenguajes más utilizados para generar *scripts* CGI son Perl y C. El primero cuenta con operadores y funciones integradas que hacen muy fácil y rápida la realización de la mayoría de las tareas relacionadas con el procesamiento de texto. En lo que respecta al lenguaje C, este dispone de operaciones a nivel sistema que permiten crear aplicaciones rápidas y poderosas usando un mínimo de recursos.

Los *scripts* CGI son almacenados en el directorio *cgi-bin* del servidor. De forma general, los pasos para la ejecución de un *script* CGI son: 1) los navegadores hacen una petición al servidor, 2) este genera procesos hijos para cada uno de ellos, 3) cada navegador envía datos a los *scripts* correspondientes, 4) estos son ejecutados y finalmente 5) los resultados se envían a la salida estándar para ser transferidos a los navegadores correspondientes.

4.3.2. Evolución de los lenguajes para en el desarrollo de aplicaciones Web

Inicialmente, la Web era sólo una colección de páginas estáticas, que podía ser consultada y/o descargada. El lenguaje HTML sirve para la generación de páginas estáticas, ya que a través de él se pueden visualizar documentos, imágenes, sonidos y otros elementos multimedia. Posteriormente se buscó y se desarrolló un método para construir páginas dinámicas, las cuales actualizan su contenido durante el proceso de despliegue.

Una de las primeras formas que se encontraron para dar dinamismo a las páginas HTML fue la metodología CGI *(Common Gateway Interface)*. Esta metodología [Rowe, 1996] define un mecanismo mediante el cual se crean pequeños programas que se ejecutan en el servidor. De esta manera, se puede aportar contenido dinámico mediante el paso de información entre el servidor HTTP y dichos programas. El resultado es un código HTML que se incluye en la página Web justo antes de ser enviada al cliente. Pese a que un *script* CGI es fácil de utilizar, en general es ineficiente porque cada vez que un cliente solicita una página a algún *script* CGI, el servidor tiene que cargarlo en memoria para su ejecución. Dicha situación ocasiona un tiempo de espera elevado. El número de usuarios también tiene repercusiones en el desempeño del *script* CGI ya que por cada usuario agregado, el requerimiento de memoria se incrementa porque todos los procesos deben ser cargados para su ejecución.

Una alternativa a CGI fue ISAPI *(Internet Server Application Programming Interface)*. Esta API [Rowe, 1996] proporciona la funcionalidad necesaria para construir

una aplicación servidora de Internet. A diferencia de CGI, que trabaja sobre archivos ejecutables, ISAPI trabaja sobre archivos DLL. Esta diferencia hace que ISAPI sea un sistema más rápido, ya que por tratarse de una biblioteca dinámica sólo será cargada una vez y podrá ser compartida por múltiples procesos, lo que supone pocos requerimientos de memoria.

Posteriormente estas técnicas fueron sustituidas por la incorporación de secuencias de órdenes (*scripts*) ejecutadas directamente en el interior de la página HTML. En lugar de consultar al servidor acerca de un ejecutable, el navegador puede ahora procesar las secuencias de órdenes a medida que carga la página HTML. El tratamiento de estas secuencias puede hacerse tanto en el servidor como en el cliente. Los lenguajes más comunes para la escritura de secuencias de órdenes son JavaScript y VBScript.

Apoyándose en la técnica anterior y en un intento de potenciar la inclusión de contenido dinámico en páginas Web, Microsoft lanza una nueva tecnología, las páginas ASP *(Active Server Page* – Página activa del servidor o activada en el servidor). Una página ASP, dicho de una forma sencilla, es un fichero .asp que puede contener texto, código HTML, secuencias de órdenes y componentes ActiveX. Mediante esta combinación, se puede conseguir de una forma muy sencilla páginas dinámicas y aplicaciones muy potentes para la Web.

Cuando un cliente solicita una ASP, el servidor la busca dentro del directorio solicitado al igual que sucede con las páginas HTML. Si es encontrada se ejecutarán las rutinas VBScript o JScript que contenga; el resultado estará dado en un formato HTML estándar. Una vez que la página ha sido procesada, el servidor enviará al cliente el contenido de la misma en HTML estándar, siendo así accesible desde cualquier navegador. Una herramienta que permite crear este tipo de páginas es Microsoft Visual Interdev. La tecnología ASP es más sencilla y potente que CGI e ISAPI.

Otra alternativa para crear páginas Web dinámicas son los *applets* Java. Esta tecnología permite vincular código estándar Java con páginas HTML que luego se utilizarán como interfaz de usuario, dotándolas de contenido dinámico e interactivo.

Por último se tiene a los *servlets*, los cuales son programas que se ejecutan en un servidor Web. Los clientes pueden invocarlos utilizando el protocolo HTTP. Una comparación entre *applets* y *servlets* sería: un *applet* es cargado y ejecutado por un navegador, mientras que un *servlet* es cargado y ejecutado por un servidor Web.

Capítulo 5

Diseño del sistema

A lo largo de este capítulo se describe el modelado del sistema propuesto, utilizando diagramas de flujo de datos. Los subsistemas representados son: 1) el contexto general del sistema, 2) la decisión del usuario sobre qué entidades desea recibir, 3) la obtención de las capacidades del sitio de trabajo (la velocidad del procesador, el espacio disponible en disco duro, el tamaño de la RAM y la resolución horizontal de pantalla), 4) la selección de versiones de las entidades, 5) la verificación de derechos de acceso y 6) el mecanismo de distribución de la red principal.

5.1. Arquitectura de distribución adaptable

Las arquitecturas básicas utilizadas en los sistemas colaborativos son la arquitectura centralizada y la arquitectura replicada (ver capítulo 2). Tomando como base dichas arquitecturas se puede generar otras, entre las que se encuentran: la arquitectura híbrida, la arquitectura asimétrica y la arquitectura de múltiples servidores.

Una distribución no adaptable de las entidades compartidas queda establecida al inicio de una sesión colaborativa y permanece durante el transcurso de dicha sesión. Por el contrario distribución adaptable de las entidades compartidas puede variar durante el transcurso de una sesión colaborativa dependiendo de varios factores: la decisión del usuario, las capacidades de los sitios participantes, los derechos de acceso y la existencia de software que procese a las entidades en el sitio de trabajo.

Una limitación importante de la **distribución no adaptable** es que el sitio participante podría no recibir la entidad compartida, e.g., cuando las capacidades del sitio no den soporte de almacenamiento o de procesamiento a dicha entidad (ver figura 5.1).

Un ejemplo de **distribución adaptable** se puede observar en la figura 5.2. En la parte izquierda está representado un sitio de almacenamiento (servidor) y su contenedor de entidades compartidas en el cual existe un ejemplar de cada entidad, además de varios representantes (mensajes que indican algún problema). Dependiendo de las capacidades del sitio de trabajo, se puede definir un modelo de arquitectura de dis-

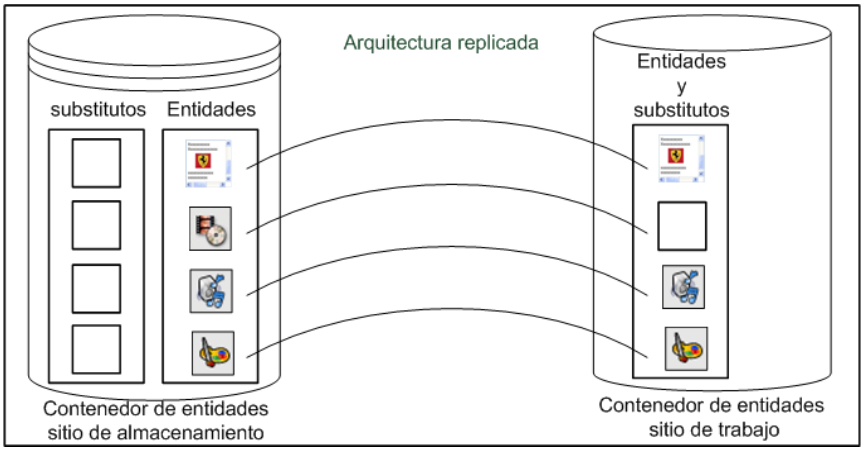

Figura 5.1: Distribución no adaptable

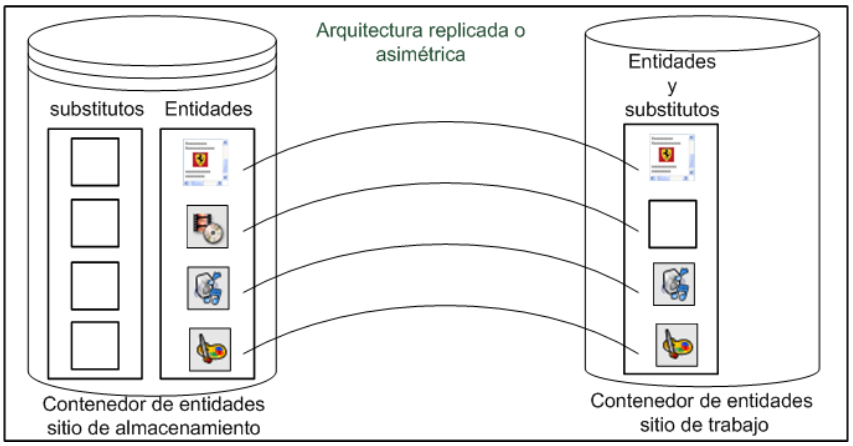

Figura 5.2: Distribución adaptable

tribución para las entidades compartidas, de las cuales se transmite un solo ejemplar al sitio de trabajo (cliente).

Considerando todo lo mencionado anteriormente, se diseñó una arquitectura de distribución adaptable que busca a la representación de la entidad compartida más adecuada en base a los siguientes parámetros: 1) la decisión del usuario, 2) las capacidades del sitio de trabajo, 3) la existencia del software que procese dicha entidad en el sitio de trabajo y 4) los derechos de acceso del usuario.

La arquitectura se puede dividir en dos enfoques:

- arquitectura de distribución adaptable para los sitios de almacenamiento,
- arquitectura de distribución adaptable para los sitios participantes.

El primero es hacia los sitios de almacenamiento que integran la red principal de

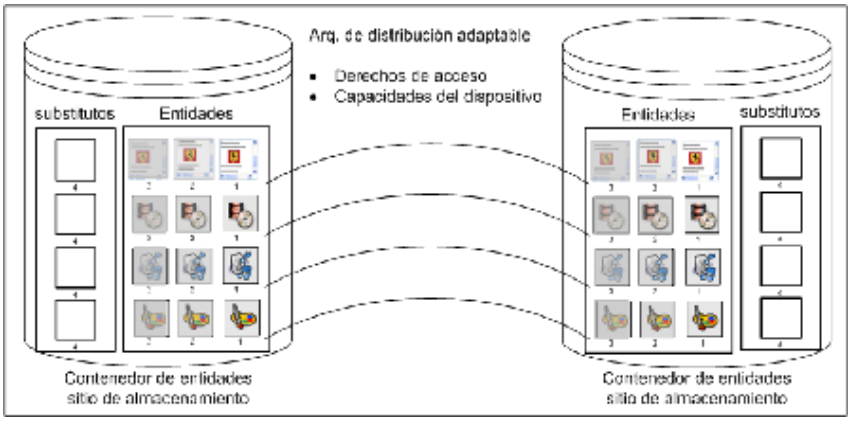

Figura 5.3: Arquitectura de distribución adaptable entre sitios de almacenamiento

distribución del sistema y el otro es hacia las subredes, las cuales están formadas por un sitio de almacenamiento y todos los sitios participantes conectados a este.

En la figura 5.3 se muestra la distribución y el almacenamiento de las entidades compartidas entre los sitios de almacenamiento. Como puede observarse, cada sitio de almacenamiento dispone de un contenedor de entidades. Por cada una de las entidades compartidas existen tres versiones, la primera es la original, la segunda corresponde a una versión degradada en el caso de las imágenes o en el caso de texto a un resumen del archivo original, y la tercera corresponde a una versión aún más degradada o a una síntesis, además de los substitutos que se utilizarán cuando el dispositivo del usuario no cubra con los requerimientos mínimos en capacidades. Cada versión de una entidad compartida no significa que está degradada en un determinado porcentaje, más bien es una versión con ciertas características determinadas y reducidas en base a las necesidades a satisfacer.

La distribución de las entidades compartidas está dada por los parámetros de derechos de acceso y las capacidades del dispositivo. Las capacidades corresponden a 1) la velocidad del procesador, 2) el espacio disponible en el disco duro, 3) el tamaño de la memoria RAM y 4) la resolución horizontal de la pantalla.

En la figura 5.4 se muestra la ubicación de las versiones de las entidades compartidas en los sitios de almacenamiento y de trabajo. En el sitio de almacenamiento existen tres versiones de cada entidad, además de un substituto. Los parámetros que se toman en cuenta para la distribución de las entidades son las capacidades del dispositivo receptor, los derechos de acceso sobre las entidades compartidas, la decisión del usuario sobre recibir o no cada una de las entidades que tiene asignadas y la existencia del software que de soporte al procesamiento de dichas entidades. Al sitio de trabajo se le transmite cada una de las entidades compartidas y los subtitutos necesarios. Tanto entidades como substitutos son almacenados en un directorio local específico.

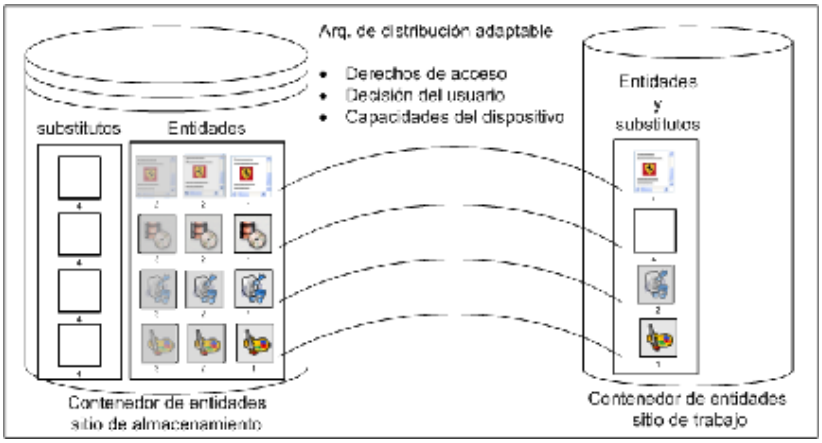

Figura 5.4: Arquitectura de distribución adaptable para los sitios participantes

5.2. Principio de autor multi-sitios

Como este trabajo de tesis se inscribe en el marco del proyecto PIÑAS *("Platform for Interaction, Naming and Storage")*, la arquitectura de distribución prevista tomará como base el principio de autor multi-sitios [Decouchant et al., 2001] [Morán et al., 2002]. Este principio constituye una de las guías de diseño de la plataforma PIÑAS para facilitar la cooperación entre personas físicamente distribuidas y potencialmente móviles. Esencialmente, este principio ofrece una organización de sitios que permite a los usuarios beneficiar de una alta disponibilidad de las entidades compartidas, aún cuando estén inmersos en un entorno poco confiable como el Internet. Esta organización de sitios también ofrece a los colaboradores la posibilidad de poder trabajar en diferentes lugares físicos de manera que, cuando se desplacen entre ellos, los colaboradores podrán recuperar automáticamente su entorno de trabajo. El principio de autor multi-sitios es descrito en la sección 5.2.1

Con el fin de caracterizar los sitios participantes en una sesión de colaboración, el principio de autor multi-sitios hace la distinción entre el lugar de procesamiento y el lugar de almacenamiento de las entidades compartidas. Esta distinción permite soportar no sólo aplicaciones PIÑAS, sino también aplicaciones Web estándar: el lugar de procesamiento y el lugar de almacenamiento corresponden respectivamente al lugar de uso y al lugar de almacenamiento, identificados por Ramduny y Dix en su análisis de arquitecturas para aplicaciones colaborativas en la Web [Ramduny and Dix, 1997].

El lugar de procesamiento se refiere al espacio donde las entidades son temporalmente memorizadas con el objeto de ser accedidas (para fines de consulta o modificación) por los colaboradores durante la sesión de trabajo en curso. El lugar de almacenamiento se refiere al espacio donde las entidades son memorizadas de manera permanente, con el fin de ser accesibles en sesiones de trabajo. De hecho, los lugares de procesamiento y de almacenamiento constituyen espacios lógicos que se concretizan

respectivamente en sitios de trabajo y sitios de almacenamiento.

En estos términos, un sitio de trabajo es el lugar de ejecución de aplicaciones colaborativas y aplicaciones Web estándares (e.g., navegadores y editores) que permiten a los colaboradores acceder a las entidades compartidas para consultarlas o modificarlas. Un sitio de almacenamiento está constituido de (al menos) un servidor Web que ejecuta servicios dedicados a la administración de entidades compartidas. Con el objeto de facilitar la cooperación entre personas físicamente distribuidas y potencialmente móviles, el principio de autor multi-sitios asocia varios sitios de trabajo y de almacenamiento a cada colaborador.

El principio de autor multi-sitios no impone restricciones sobre la exclusividad de uso de un sitio, ya que un mismo sitio de trabajo o de almacenamiento puede ser compartido por varios colaboradores. Asimismo, este principio tampoco impone restricciones sobre la exclusividad del rol que juega cada sitio, puesto que un mismo sitio puede actuar simultáneamente como sitio de trabajo y como sitio de almacenamiento. De esta manera, es posible soportar el trabajo en modo temporalmente desconectado o autónomo.

Asimismo, el principio de autor multi-sitios también define asociaciones entre los sitios de trabajo y los sitios de almacenamiento disponibles a un colaborador. Estas asociaciones constituyen un primer paso hacia el soporte del trabajo nómada, ya que un colaborador puede establecer conexiones desde y hacia sitios predefinidos a medida que se desplaza. Además, estas asociaciones entre sitios permiten realizar optimizaciones (e.g., acceso a las entidades desde un sitio próximo) y validaciones (e.g., identificación de los sitios que necesitan ciertas entidades). De esta manera, un colaborador no sólo dispone de un acceso transparente a las entidades compartidas a partir de sitios diferentes, sino también puede recuperar dinámicamente y en cualquier momento su espacio de trabajo.

Ejemplo

Con el fin de ilustrar el principio de autor multi-sitios, suponga que los colaboradores potenciales *Maggie, Lisa, Homer y Bart*, dispersos alrededor del mundo, tienen a su disposición varios sitios. *Bart* puede consultar y modificar entidades compartidas desde los sitios de trabajo *chichenitza.unam.mx*, *versailles.cnrs.fr* y *neushwanstein.gmd.de* (cf. Figura 5.5). Igualmente, *Bart* puede cargar/descargar entidades compartidas desde/hacia los sitios de almacenamiento *uxmal.unam.mx*, *chambord.cnrs.fr, blois.cnrs.fr y linderhof.gmd.de*. Tanto los sitios de trabajo como los sitios de almacenamiento de *Bart* pertenecen a diversas instituciones y están localizados en diferentes países (la UNAM en México, el CNRS en Francia y el GMD en Alemania).

La organización de sitios precedente ofrece a *Bart* la posibilidad de trabajar en modo nómada. Dependiendo de la localización actual de *Bart*, e.g., chichenitza.unam.mx, él accede a las entidades administradas por el sitio de almacenamiento asociado más próximo, e.g., *uxmal.unam.mx*. De esta manera, *Bart* puede a la vez desplazarse de un punto a otro del Internet y disponer de un entorno de trabajo distribuido más confiable, similar al que ofrece una LAN.

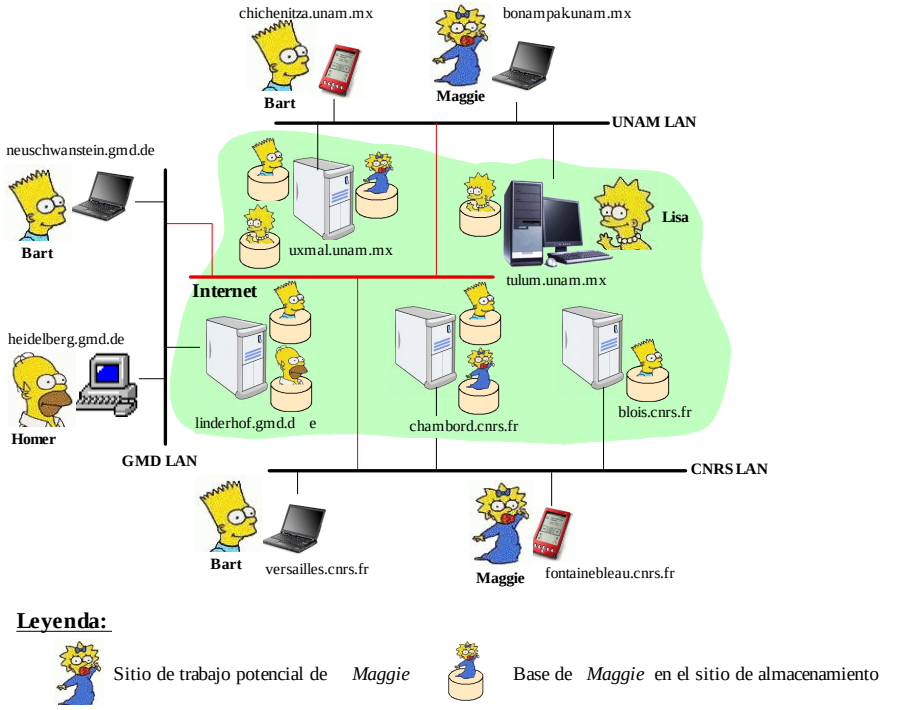

Figura 5.5: Organización de sitios según el principio de autor multi-sitios

Por otra parte, los sitios de trabajo de los diferentes colaboradores, e.g., el sitio *fontainebleau.cnrs.fr* de *Maggie* y el sitio *versailles.cnrs.fr* de *Bart* pueden cargar y descargar entidades desde/hacia un sitio de almacenamiento común, e.g., *chambord.cnrs.fr*. De igual manera, el sitio *bonampak.unam.mx* de *Maggie*, el sitio *tulum.unam.mx* de *Liza* y el sitio *chichenitza.unam.mx* de *Bart* utilizan los servicios ofrecidos por un mismo sitio, e.g. *uxmal.unam.mx*. Así, varios colaboradores cuyos sitios de trabajo están localizados en lugares relativamente cercanos, e.g., una LAN, pueden compartir las capacidades de almacenamiento y procesamiento de un mismo sitio.

También es posible que uno de los sitios de trabajo de un colaborador, e.g., el sitio *versailles.cnrs.fr* de *Bart*, haga uso exclusivo de un sitio de almacenamiento, e.g., *blois.cnrs.fr*. De hecho, cuando *Bart* trabaja en el sitio *versailles.cnrs.fr*, por diversas razones (indisponibilidad de uno de los sitios, carga fuerte de un sitio o lentitud de una conexión de red) el sistema puede privilegiar un sitio de almacenamiento dado, e.g., las entidades accedidas por *Bart* pueden ser cargadas/descargadas desde/hacia *chambord.cnrs.fr* o *blois.cnrs.fr*.

Sin querer detallar el entorno de trabajo de *Homer*, note que únicamente puede trabajar en el sitio *heidelberg.gmd.de* accediendo las entidades desde el sitio de almacenamiento *linderhof.gmd.de*. Finalmente, se puede observar que el sitio *tulum.unam.mx* es capaz de actuar simultáneamente como sitio de trabajo y como sitio de almace-

namiento. Así, *Liza* puede trabajar de manera completamente autónoma, i.e. sus sesiones de trabajo se vuelven insensibles a las perturbaciones potenciales de la red.

5.3. Modelado del sistema de distribución adaptable

El modelado del sistema de distribución adaptable se realiza mediante diagramas de flujo de datos [Pressman, 2002]. Un diagrama de flujo de datos es un modelo lógico gráfico para representar el funcionamiento de un sistema en un proyecto de software. Sus elementos gráficos son:

- **rectángulos cerrados**: representan entidades externas,
- **rectángulos abiertos**: describen almacenes o archivos,
- **círculos**: significan procesos,
- **flechas**: flujos de datos desde o hacia un proceso.

Un diagrama de flujo de datos puede ser profundizado expandiendo algunos de sus procesos en subprocesos; en este caso la etiqueta tendrá un número adicional. No hay límite en el número de procesos. Los diagramas derivados de los procesos principales se clasifican en tres niveles:

- **Diagrama de contexto (nivel 0):** es este diagrama se modela el proceso principal del problema en cuestión con sus respectivas entidades. Cada proceso debe tener al menos una entrada y una salida de datos.
- **Diagrama de nivel superior (nivel 1):** en el diagrama de nivel superior se plasman todos los procesos que describen al proceso principal o diagrama de contexto. En este nivel aparecen los archivos, los cuales se utilizan para almacenar datos para que sean usados en otros procesos como entradas ya sea para configuración o para realizar una determinada tarea, además muchas de las salidas de estos procesos se almacenan en archivos.
- **Diagrama de detalle o expansión (nivel 2):** en este diagrama se expanden cada uno de los procesos del nivel anterior con el objeto de definir y entender mejor el sistema que se está modelando. Los diagramas de nivel superior y de detalle deben tener las entradas y salidas definidas en el diagrama de contexto.

El sistema de distribución adaptable está dividido en varios subsistemas:

1. diagrama de contexto,
2. decisión del usuario,

3. obtención de capacidades,
4. adaptación a dispositivos,
5. adaptación a derechos de acceso,
6. diagrama de la red principal de distribución. En las siguientes subsecciones se muestran gráficamente dichos subsistemas.

5.3.1. Diagrama de contexto

La estructura principal del sistema de distribución adaptable está formado por un grupo de sitios de almacenamiento (red principal). Cada sitio integrante del grupo de almacenamiento está asociado a varios sitios de trabajo (ver figura 5.6), lo cual origina una subestructura de colaboración entre sitios (sub-redes). Cada sitio de trabajo se comunica con el sitio de almacenamiento local para solicitar las entidades compartidas requeridas por el usuario que ha iniciado sesión. Dichas entidades pueden ser de lectura, de escritura y nulo (acceso denegado).

Respecto al sitio de trabajo, las entradas y salidas del sistema están representadas por: el teclado, la pantalla y la conexión de red. La entidad externa está dada por la base de datos local de las entidades. En lo referente al sitio de almacenamiento, las entidades externas son dos bases de datos: una para los usuarios y otra para las entidades compartidas. Por último los dispositivos de entrada y salida están representados por la interfaz de red.

5.3.2. Decisión del usuario

En la figura 5.7 se muestra el mecanismo para permitir que el usuario decida qué entidades compartidas desea recibir. El funcionamiento inicia cuando el usuario ha sido autenticado en el sitio de almacenamiento. El primer proceso en ejecutarse sirve para `obtener el listado de entidades asociadas al usuario`, en tanto que el siguiente proceso en activarse permite `obtener el listado de entidades existentes`. Los dos procesos anteriores sirven de entrada al tercer proceso, el cual `genera el listado de entidades asociadas al usuario y en existencia` en el sitio de almacenamiento. Posteriormente, este listado es enviado al sitio de trabajo para que el usuario pueda elegir las entidades compartidas que desea recibir. Dichos eventos suceden mediante los procesos `enviar y recibir listado de entidades`; a continuación se `despliegan y se captan las entidades elegidas` por el usuario. El siguiente proceso en llevarse acabo `genera el listado de entidades aceptadas.` Por último, se inicia nuevamente el `envío y recepción de listado` entre el sitio de trabajo y el sitio de almacenamiento. Como se puede observar este mecanismo tiene tres flujos de datos a otros mecanismos representados por los números 1, 2 y 3: el primero se dirige al `mecanismo de adaptación a derechos de acceso`, el segundo hacia el `mecanismo de obtención de capacidades` y el tercero al `mecanismo de adaptación a dispositivos`. Dichos mecanismos serán explicados posteriormente.

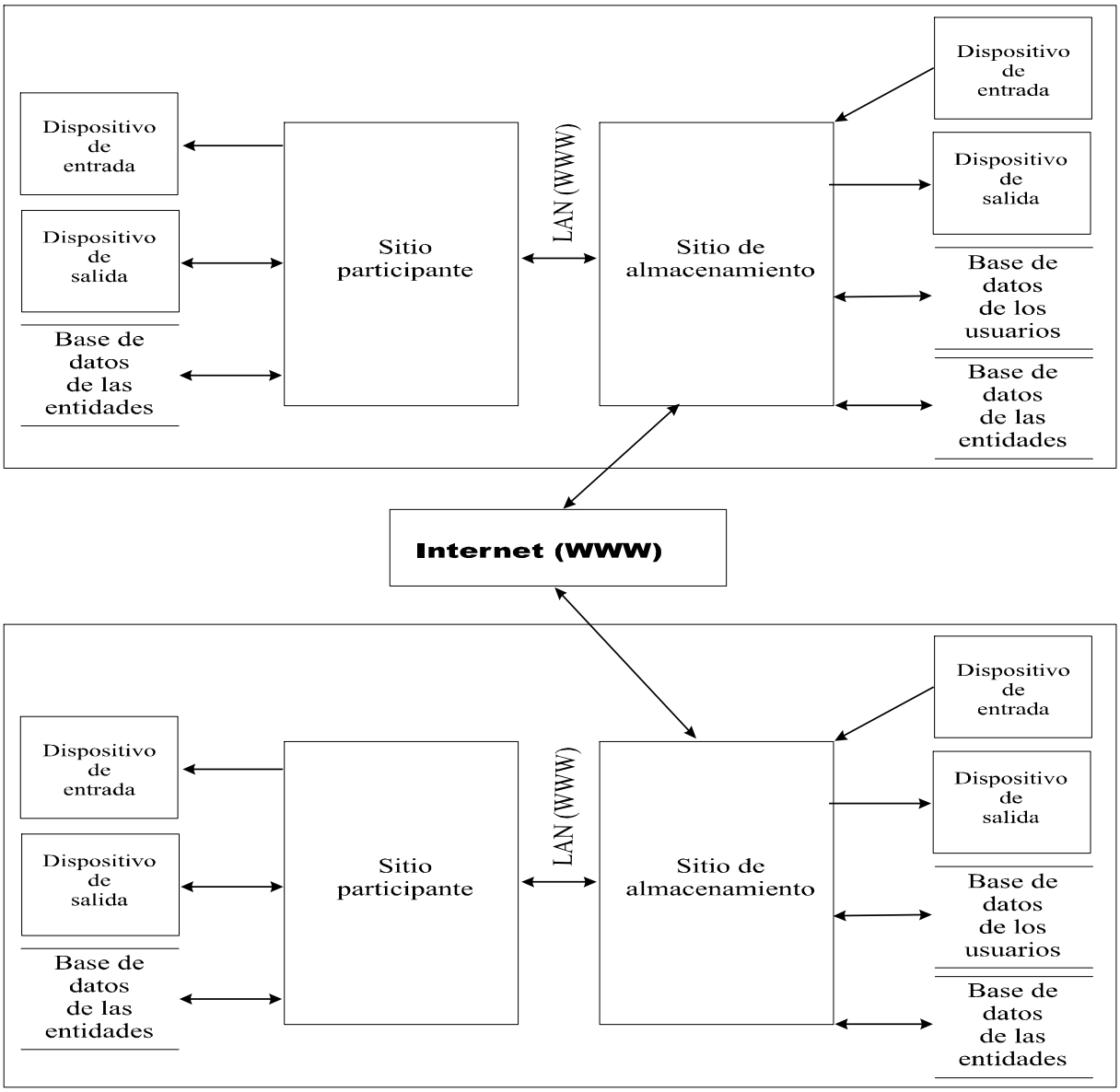

Figura 5.6: Diagrama de contexto

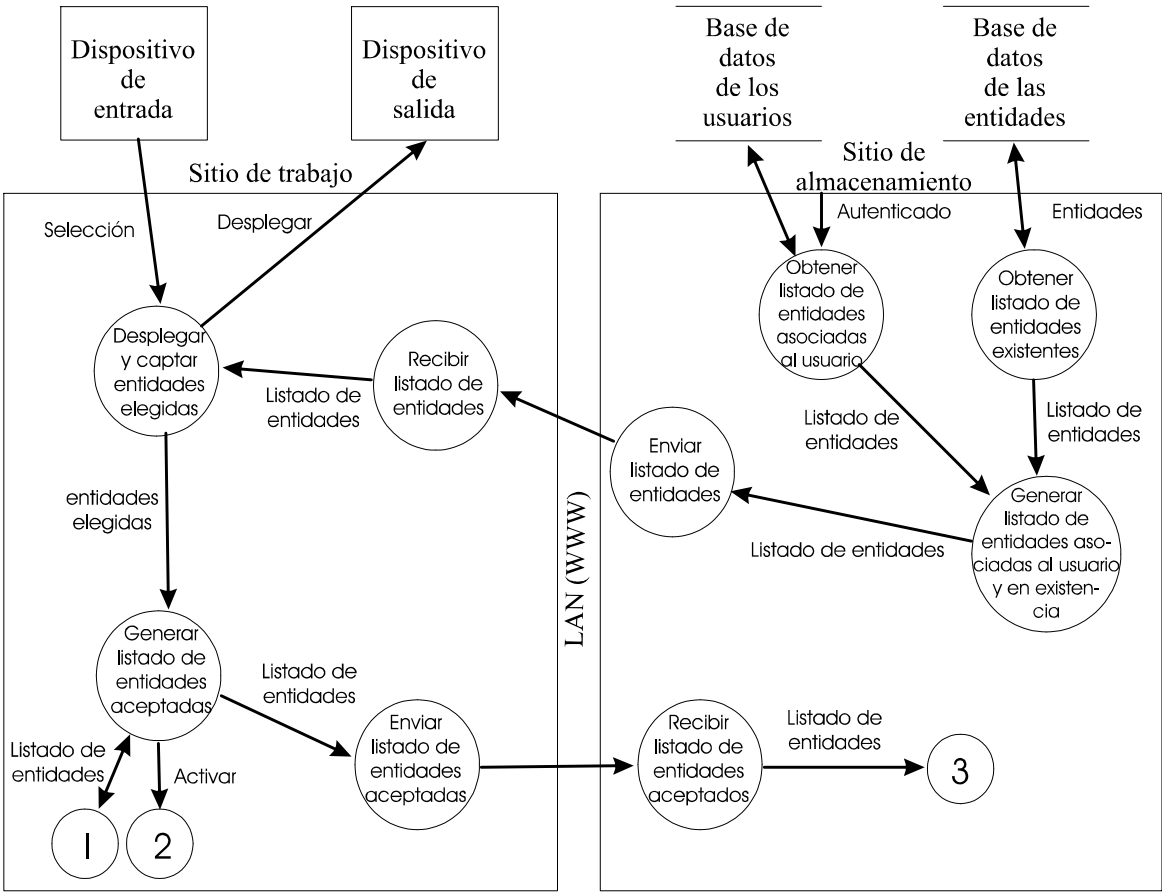

Figura 5.7: Diagrama de decisión del usuario

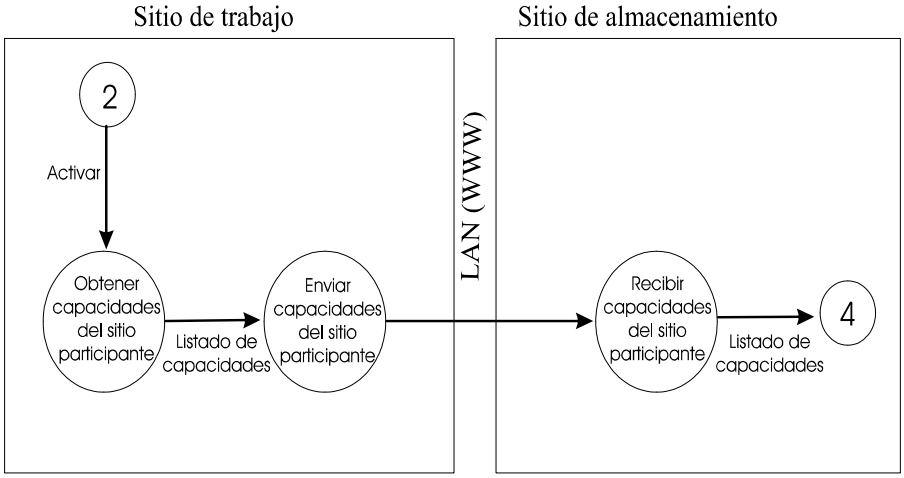

Figura 5.8: Diagrama de obtención de capacidades

5.3.3. Obtención de capacidades

Este mecanismo obtiene las capacidades (variables lingüísticas) del sitio de trabajo (ver figura 5.8), las cuales son: 1) la velocidad del procesador, 2) el espacio disponible del disco duro al momento de la solicitud de las entidades, 3) el tamaño de la memoria RAM y 4) la resolución horizontal de la pantalla.

El mecanismo se activa cuando el proceso `generar listado de entidades` aceptadas, perteneciente al `mecanismo de decisión del usuario`, ha terminado su función. A continuación, se ejecuta el proceso para `obtener las capacidades del dispositivo`, las cuales son guardadas en un archivo que es transferido al sitio de almacenamiento para su posterior utilización en el mecanismo de adaptación (número 4 de la figura 5.8).

5.3.4. Selección de versiones de las entidades

La funcionalidad de este mecanismo es determinar las versiones más adecuadas de las entidades compartidas para el sitio de trabajo en base a varios factores que serán explicados a continuación.

Para que este mecanismo funcione adecuadamente se debe cubrir cuatro requisitos: 1) la existencia del listado de las entidades aceptadas, 2) la existencia del listado de las capacidades del dispositivo, 3) la definición de los límites para los conjuntos difusos y 4) la existencia del listado con las extensiones tipo y subtipo MIME soportadas por el sitio participante.

Una vez cubiertas dichas condiciones entra en funcionamiento el mecanismo de selección de versiones de las entidades (ver número 3 en la figura 5.9). `El listado de las entidades aceptadas por el usuario` es cargado al sistema. Posteriormente se `envía uno de los registros del listado` de las `entidades aceptadas` para

`su evaluación` al siguiente proceso, además se debe `cargar el archivo de` extensiones al sistema. A continuación se determina si `existe algún software` que soporte la entidad evaluada para asignarlo a alguna de las cinco categorías multimedia definidas: 1) texto, 2) imagen, 3) animación, 4) audio y 5) video.

El sistema también debe cargar `las capacidades del dispositivo` (ver número 4 en la figura 5.9). Dichas capacidades son pasadas al proceso de fuzzificación, en el que ya están definidos los límites de los conjuntos difusos. Otra de las entradas que debe tener el proceso de fuzzificación es el `tamaño de la entidad a evaluar`. Al finalizar este proceso se arroja como resultado cinco valores difusos correspondientes a las cuatro capacidades del sitio de trabajo y al tamaño de la entidad evaluada.

El siguiente proceso ejecutado es el `mecanismo de inferencia`, el cual recibe los cinco valores difusos y la categoría multimedia a la que corresponde la entidad evaluada. Dicho proceso produce como resultado la versión adaptada de la entidad, la cual es almacenada en el registro original de esta.

Los pasos descritos anteriormente sólo evalúan una entidad, por tal motivo se tiene que aplicar un ciclo en el que se evalúe el siguiente registro del listado de entidades aceptadas y así consecutivamente hasta cubrir todos los registros.

El mecanismo de distribución utiliza el listado de entidades adaptadas para realizar el envío de las entidades compartidas, las cuales están ubicadas en un contenedor en el sitio de almacenamiento (ver número 3 en la figura 5.9).

5.3.5. Verificación de derechos de acceso

Este mecanismo está ubicado en su totalidad en el sitio de trabajo, el cual recibe el `listado de las entidades` por parte del proceso `generar listado de entidades aceptadas`, ubicado en el mecanismo de `decisión del usuario` (ver número 1 de la figura 5.10). Además del listado, este mecanismo también recibe las `entidades adaptadas` que han sido enviadas desde el `sitio de almacenamiento` por el subsistema de `adaptación a dispositivos` (ver número 5 de la figura 5.10). El listado es tratado por el proceso `extraer derechos de acceso`, el cual se encarga de extraer los derechos de acceso de cada entidad para su transferencia al proceso `aplicar derechos de acceso a cada entidad recibida`, el cual será desarrollado por el programador (ver número 2 de la figura 5.10). Estos derechos de acceso son desplegados en la pantalla y a la vez las entidades adaptadas son almacenadas en la base de datos.

5.3.6. Diagrama de la red principal de distribución

Este diagrama muestra la distribución de las entidades compartidas en el grupo principal de la red, la cual está formada por sitios de almacenamiento. El sistema entra en funcionamiento con el proceso `inicio`, a partir del cual se activan dos procesos: `obtener capacidades del sitio de almacenamiento` (ver figura 5.11) y `obtener registros de los usuarios` (ver figura 5.12). El primero obtiene las capacidades del sitio de `almacenamiento 1`, las cuales son enviadas al sitio de `almacenamiento 2`

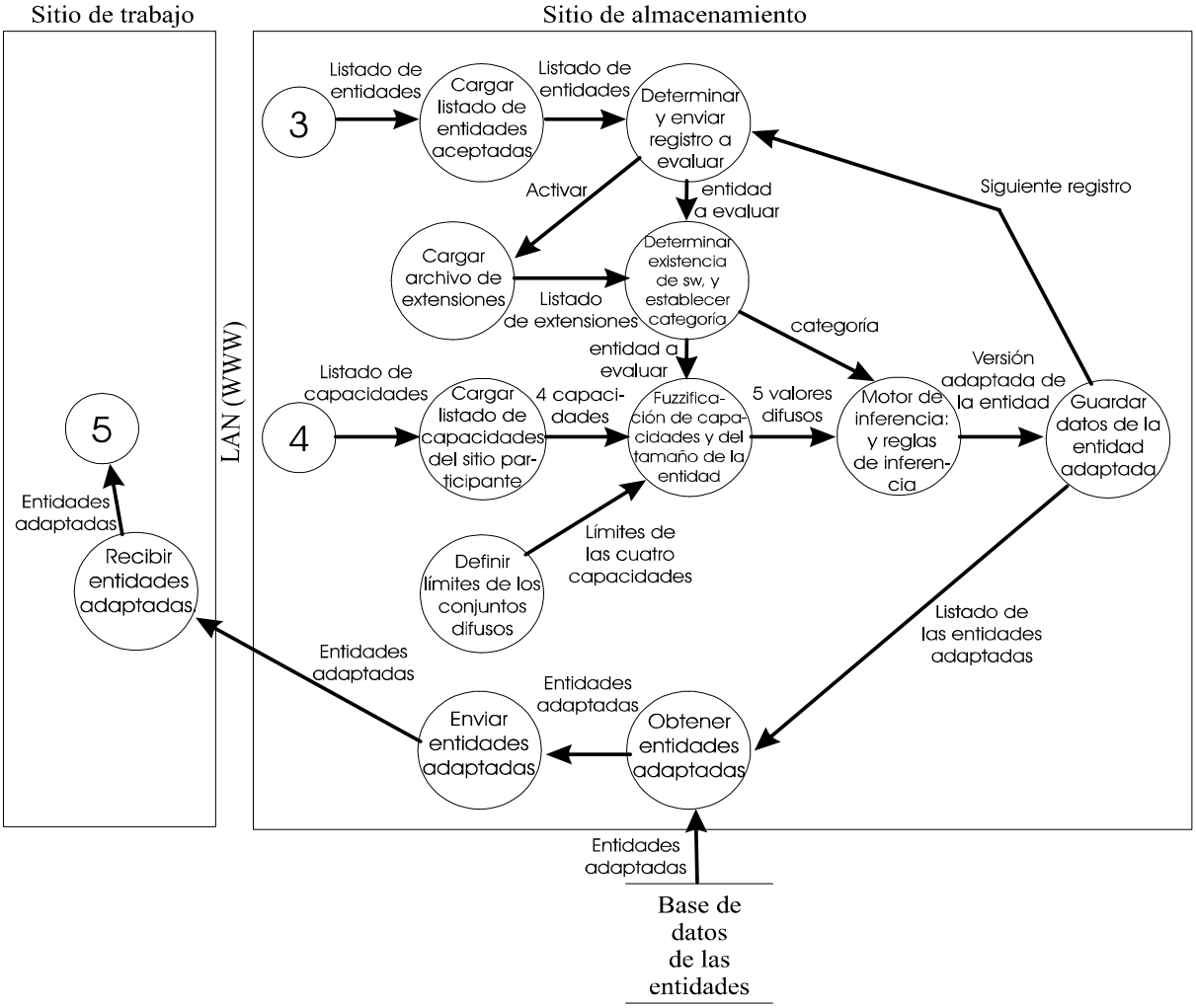

Figura 5.9: Diagrama de selección de versiones de las entidades

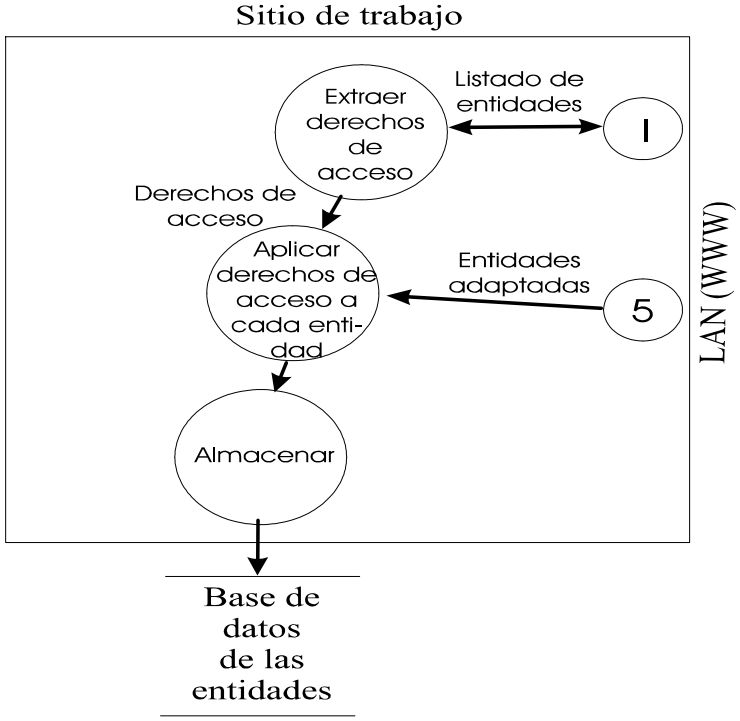

Figura 5.10: Adaptación a derechos de acceso

mediante los procesos `enviar y recibir capacidades`. El listado recibido se transfiere al proceso `determinar las entidades a enviar` al sitio de almacenamiento, pero al mismo tiempo este proceso recibe los registros de los usuarios provenientes del proceso `obtener registros de los usuarios`, el cual a su vez accede a la `base de datos de usuarios`. El proceso `determinar las entidades a enviar` genera un listado de las entidades, el cual es transferido al proceso `obtener entidades para su replicación`. Dicha obtención de entidades se logra con el acceso a la `base de datos de las entidades` para su transferencia al proceso `enviar entidades`, el cual a su vez las enviará al `sitio de almacenamiento 1`. El proceso `recibir entidades` activa el proceso `guardar entidades` en la base de datos. Considerando las entidades recibidas, se `generan y/o actualizan los registros de la base de datos de los usuarios` para saber que entidades están relacionados a cada uno de ellos.

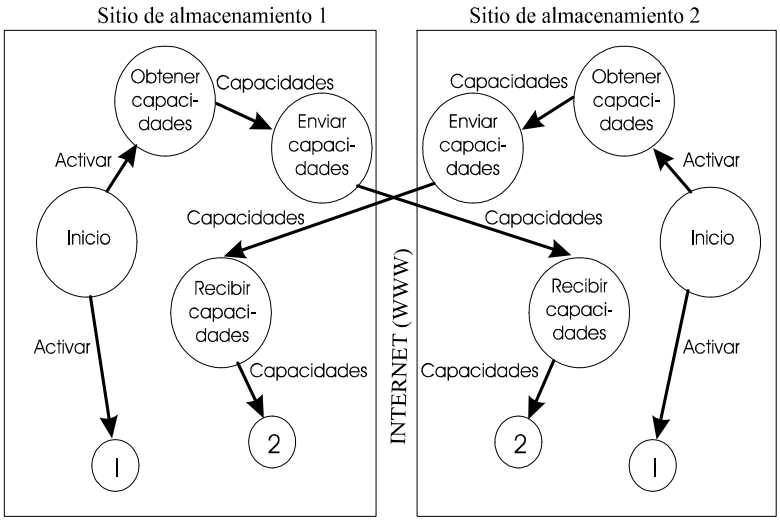

Figura 5.11: Obtener capacidades del sitio de almacenamiento

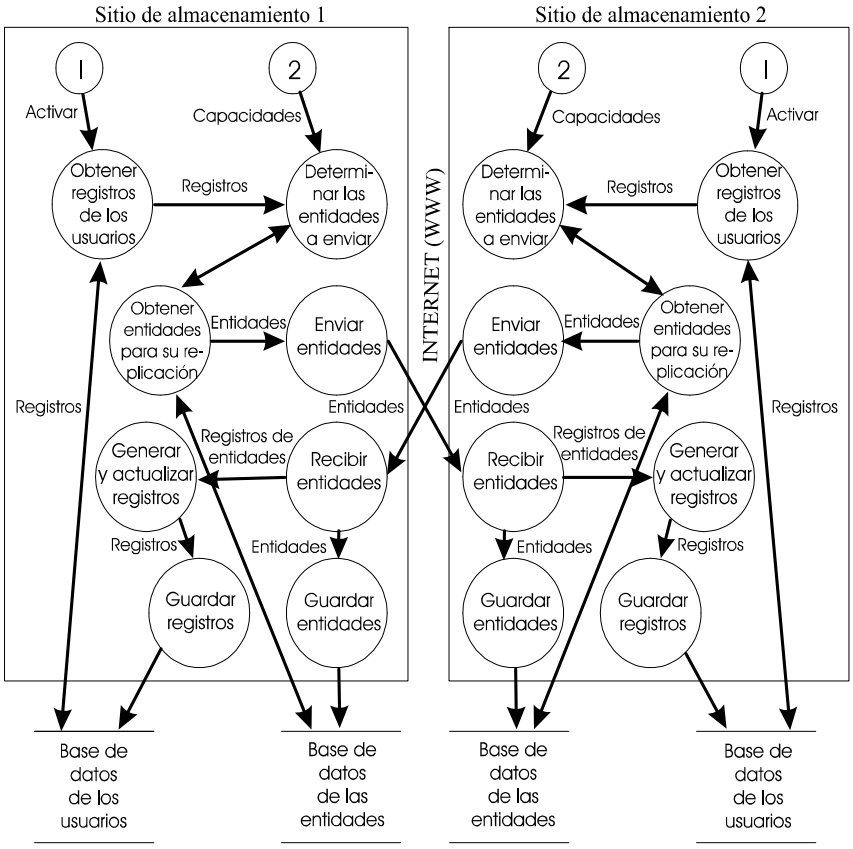

Figura 5.12: Selección de versiones de las entidades para sitios de almacenamiento

Capítulo 6

Implementación, pruebas y resultados

En el capítulo anterior se especificó el modelado de los subsistemas que integran la arquitectura de distribución adaptable de entidades: el diagrama de contexto, la decisión del usuario, la obtención de capacidades, la selección de las versiones de las entidades, la verificación de derechos de acceso y el diagrama de la red principal de distribución.

En particular en este capítulo se muestra como son obtenidas las capacidades del sitio participante, así también como se determina la versión de la entidad compartida más adecuada utilizando técnicas como la fuzzificación y el motor de inferencia diseñados especificamente para este caso. Se explica como son implementados los módulos de: la obtención de capacidades, la selección de las versiones de las entidades y el mecanismo de distribución, así como las pruebas realizadas y los resultados obtenidos.

6.1. Preliminares: obtención de capacidades

La funcionalidad de éste módulo es obtener las capacidades del sitio participante (dispositivo cliente o servidor). Dichas capacidades son: la velocidad del procesador, el espacio disponible en el disco duro, el tamaño de la RAM y la resolución horizontal de la pantalla. No se toma en cuenta la resolución vertical de la pantalla porque se incrementaría en un factor de multiplicación de tres el número de reglas de inferencia que actualmente es de 1215, dando un total de 3645. El procedimiento para la obtención de las capacidades se puede definir en los siguientes pasos (ilustrado en la figura 6.1).

1. inicialmente se definen cuatro arreglos de caracteres de tamaño cinco, los cuales en este caso se les llamó: `procesador`, `discoDuro`, `RAM` y `resolPantalla`,

2. se obtiene la velocidad del procesador en MHz y se almacena en el arreglo `procesador`,

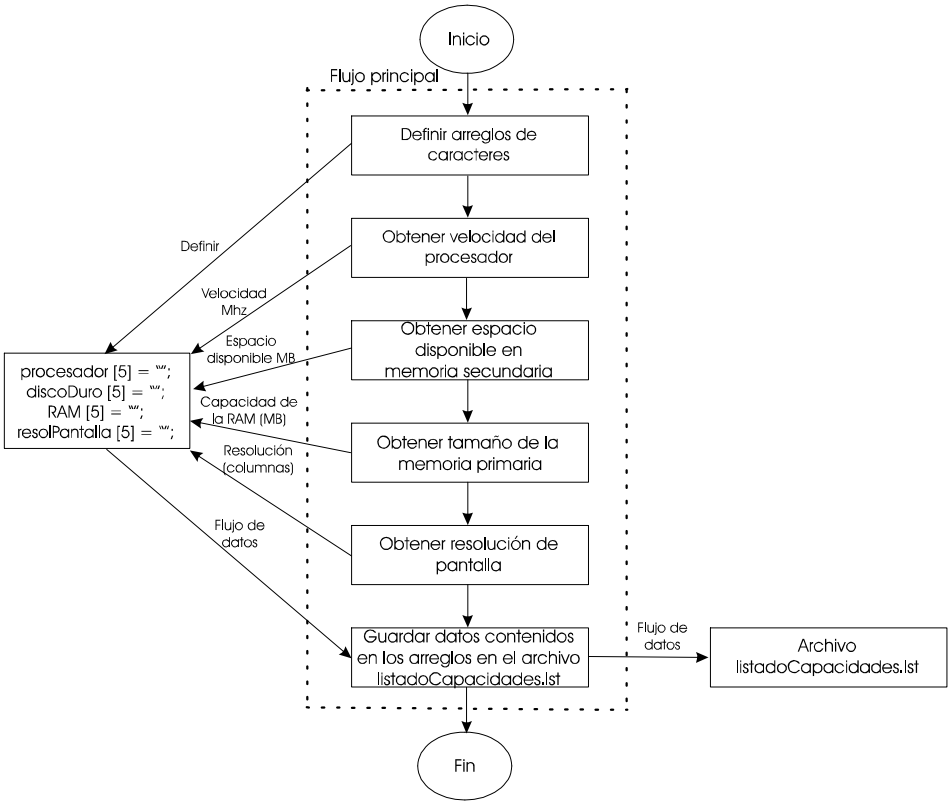

Figura 6.1: Diagrama general de la obtención de capacidades

3. se obtiene el espacio disponible en memoria secundaria en MB y se guarda en el arreglo `discoDuro`,

4. se obtiene el tamaño de la memoria primaria en MB y se almacena en el arreglo `RAM`,

5. se obtiene la resolución de pantalla (número de columnas) y se guarda en el arreglo `resolPantalla`,

6. por último se guardan los cuatro datos obtenidos en el archivo listadoCapacidades.lst.

7. para finalizar, los datos contenidos en los cuatro arreglos serán guardados en el archivo `listadoCapacidades.lst`.

En las subsecciones posteriores se explican con más detalle la obtención de las capacidades del sitio participante.

6.1.1. Definición del procesador

Para obtener la velocidad del procesador en MHz se utiliza el comando `grep` con la expresión `MHz` aplicado sobre el contenido del archivo `cpuinfo` que está ubicado en el directorio `/proc`. Este archivo está inicializado por el sistema Linux. El resultado es redireccionado al archivo `CPU` en el cual están fusionadas todas las líneas que tengan coincidencias con la expresión MHz, e.g., cpu MHz : 1596.000. Posteriormente se definen dos arreglos de caracteres llamados: `CPU` y `velocidad_MHz` para cargar los datos contenidos en el archivo `CPU`.

Como está ilustrado en la figura 6.2, se abre un flujo de solo lectura al archivo `CPU` y se cargan los valores correspondientes en dichos arreglos. A continuación se convierte a número flotante los caracteres contenidos en el arreglo `velocidad_MHz`, posteriormente este valor es convertido nuevamente a caracteres. Dicho procedimiento es para eliminar los espacios en blanco que pudiesen existir desde la obtención de los datos en el archivo. El resultado es almacenado en el arreglo `procesador`. Por último se cierra el flujo al archivo `CPU` y se elimina dicho archivo mediante la función `system ()` y el comando `rm`.

6.1.2. Información del disco duro

Para obtener el espacio disponible en el disco duro en MB se utiliza el comando `df` con el parámetro `-m`. Dicha información se redirecciona al archivo llamado `DD_AUX`. Los datos contenidos en `DD_AUX` son el tipo del sistema de ficheros, el tamaño, el espacio usado, el espacio disponible, el porcentaje de uso y la unidad de montaje. A continuación se le aplica al archivo `DD_AUX` el comando `grep` con el patrón `/home` y la salida se redirecciona al archivo `DD`. Dicho archivo contendrá información referente al directorio `home`, y estos datos serán cargados al programa en los arreglos: `sist_ficheros`, `tamaño`, `usado`, `disponible`, `uso`, `montado_en`.

El dato a utilizar es el que está contenido en el arreglo `disponible`, por tal motivo dicho dato es convertido a entero y posteriormente a caracteres. Dicho procedimiento elimina los espacios en blanco que existiesen en el arreglo `disponible`. El resultado será copiado al arreglo `discoDuro` el cual fue definido al inicio del módulo de obtención de capacidades. Por último se eliminan los archivos `DD_AUX` y `DD` (ver figura 6.3).

6.1.3. Memoria RAM

La metodología para obtener el tamaño de la memoria principal en MB es la siguiente. Se utiliza el comando `grep` con el patrón `MemTotal` sobre el archivo `meminfo` ubicado en el directorio `/proc`, cuya salida se redirecciona al archivo `MEM`.

Se definen los arreglos de caracteres `clave` y `valor` en los cuales se cargan los datos contenidos en el archivo `MEM`. El contenido del arreglo `valor` es convertido a entero y dividido entre 1024 para así realizar la conversión a MB, el resultado se convierte a caracteres y almacenado en el arreglo `RAM`. Para finalizar se elimina el archivo `MEM` (ver figura 6.4).

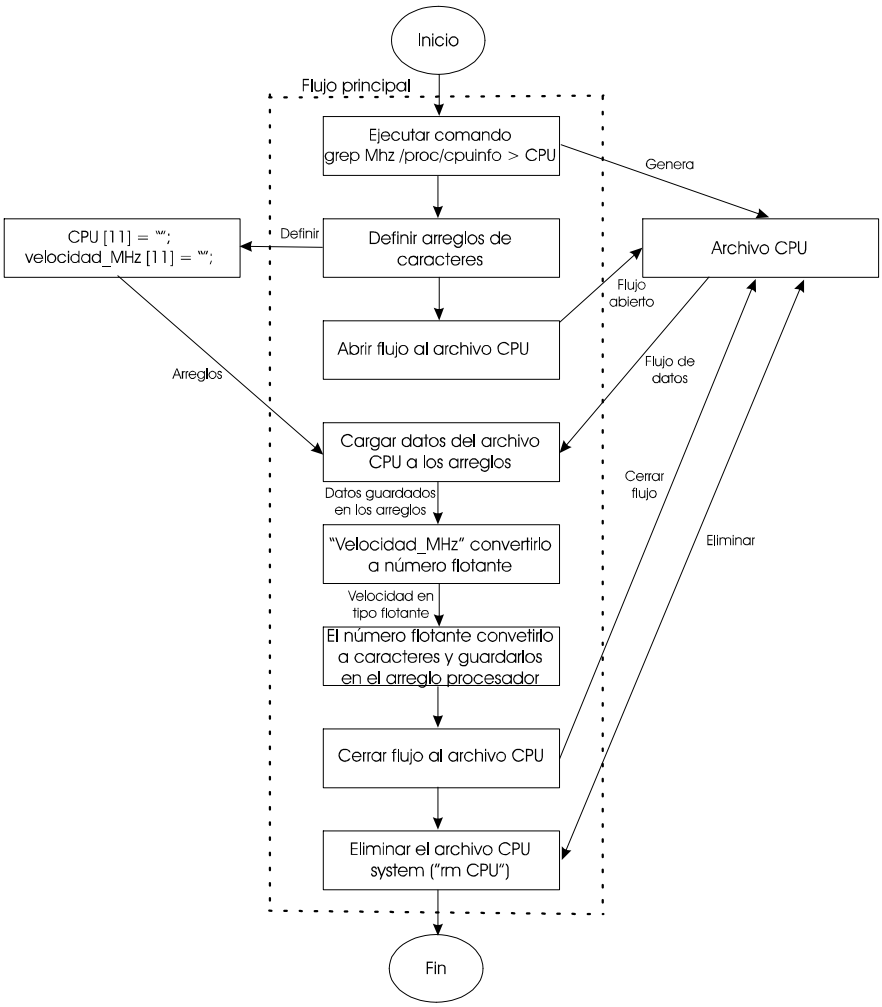

Figura 6.2: Obtención de la velocidad del procesador

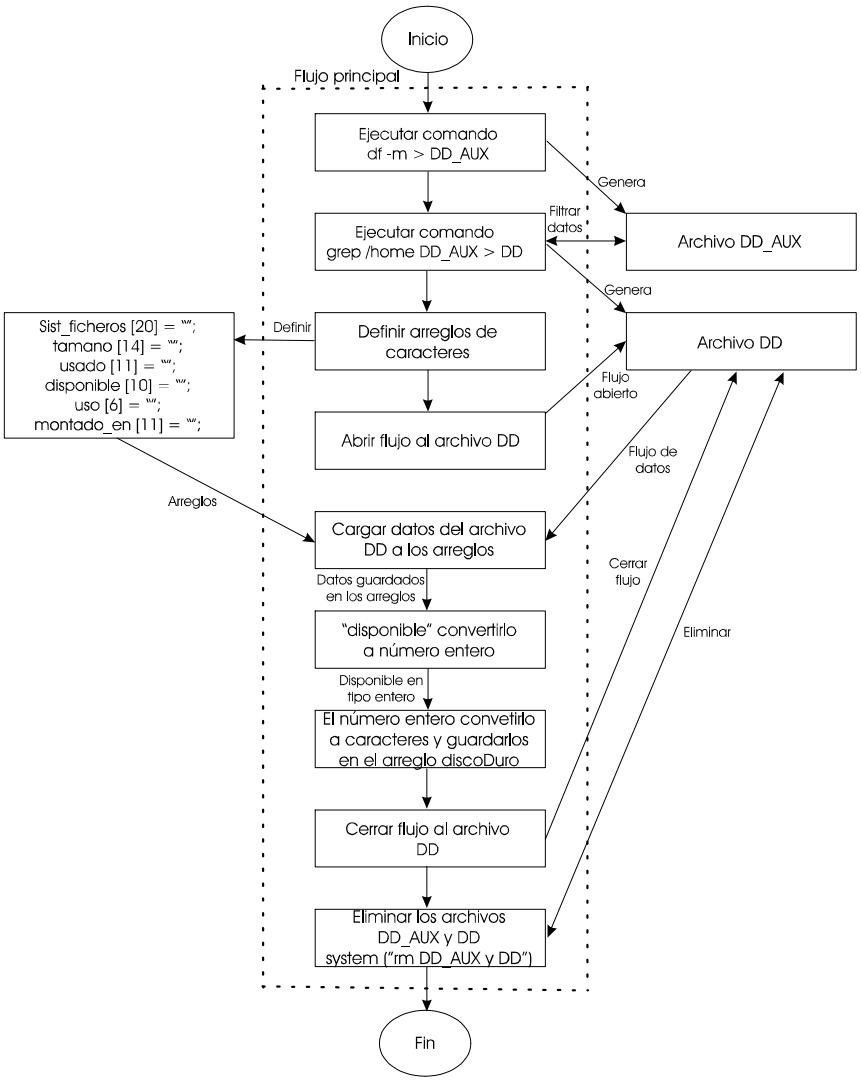

Figura 6.3: Obtención del tamaño del disco duro

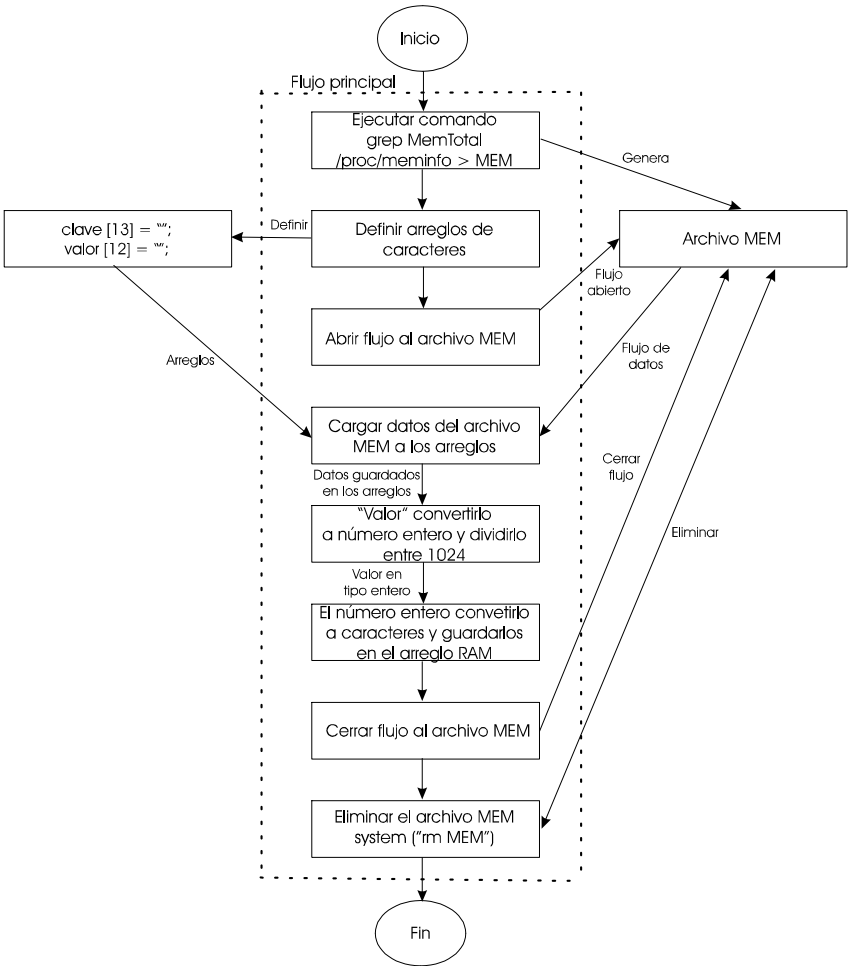

Figura 6.4: Obtención de la capacidad de la memoria RAM

6.1.4. Resolución de pantalla

La resolución de pantalla se obtiene aplicando el comando `xrandr` y cuyo resultado es redirigido al archivo `RP`. Posteriormente se aplica al archivo `RP` el comando `grep` con el patrón `"60."` y su salida se redirecciona al archivo `RESOL_PANTALLA`.

Posteriormente se abre un flujo de lectura hacia el archivo `RESOL_PANTALLA` de donde son cargados sus datos hacia los arreglos: `resolucion_X`, `X`, `resolucion_Y` y `complemento`.

El arreglo sobre el cuál se trabaja es `resolucion_X`, (representa a las columnas). El primer paso es convertir su valor a entero y posteriormente a caracteres para así eliminar los espacios en blanco que tuviese desde su extracción del archivo, el resultado se almacena en el arreglo `resolPantalla`. Por último se eliminan los archivo `RESOL_PANTALLA` y `RP`. Todo este proceso se muestra en la figura 6.5.

6.2. Adaptación a dispositivos

En este módulo se determina una de las tres versiones existentes de cada entidad compartida para ser enviada al sitio participante. Lo anterior, se lleva acabo en base a la evaluación de las capacidades de cada uno de dichos sitios (la velocidad del procesador, el espacio disponible en el disco duro, el tamaño de la memoria RAM y la resolución horizontal de pantalla), además del tamaño de cada una de las entidades compartidas solicitadas. El resultado será un listado con los datos de las entidades más adecuadas al sitio participante.

En los siguientes apartados, se explica cada uno de los submódulos que forman parte de la adaptación a dispositivos.

6.2.1. Representación gráfica de los conjuntos difusos de las capacidades de los dispositivos

Las capacidades de los sitios participantes se utilizan para determinar que entidades compartidas son más adecuadas para transmitirlas a una sesión de usuario. Dichas capacidades son representadas en el sistema mediante `variables lingüísticas` (ver subsección 3.1.2):

- procesador -> velocidad del procesador (MHz),
- disco_duro -> espacio disponible del disco duro (MB),
- RAM -> tamaño de la RAM (MB),
- resolución_de_pantalla -> resolución horizontal de pantalla (pixeles).

El `tamaño de la entidad compartida` (*Kbyte*), es otra de las variables lingüísticas para determinar las entidades compartidas adaptadas.

La estructura de los conjuntos difusos para evaluar las entidades compartidas en cada una de las variables lingüísticas son explicadas en los siguientes apartados:

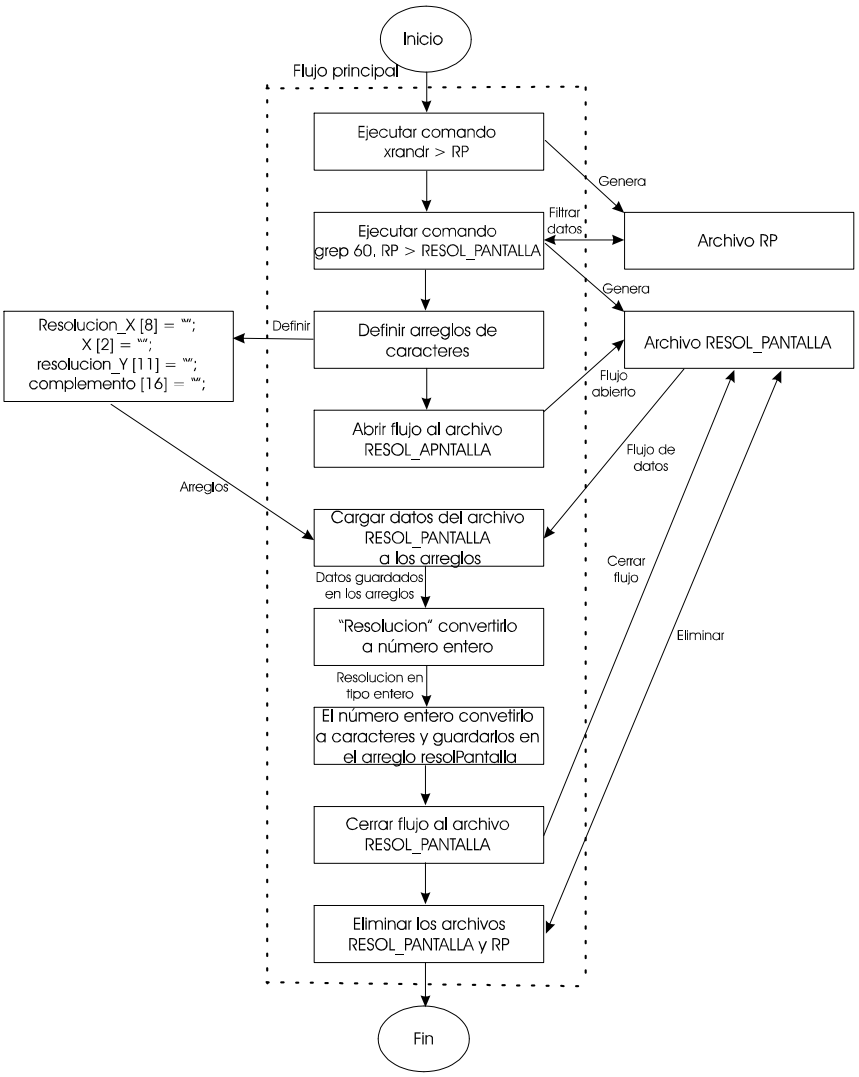

Figura 6.5: Obtención de la resolución de pantalla

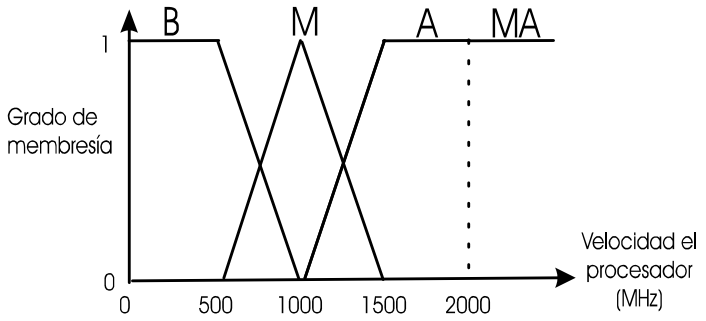

Figura 6.6: Representación gráfica difusa para el procesador

Gráfica difusa de la velocidad del procesador (MHz)

Para la evaluación de la velocidad del procesador el universo de discurso (ver subsección 3.1.2) está formado por los conjuntos `Bajo (B)`, `Medio (M)` y `Alto (A)` y la velocidad es medida en MHz de 0 a 2000. Para velocidades más altas se tiene definido el conjunto `Muy Alto (MA)`, el cual da soporte a más de 2000 MHz. En dicha situación la solución arrojada sería el conjunto **A**, porque solo se puede procesar uno de tres posibles valores en las reglas de inferencia (B, M o A)**.** En la figura 6.6 se representa el enfoque difuso de cómo se evaluá la velocidad del procesador.

Los intervalos de velocidades definidas en la gráfica 6.6 son relativos a los que se pueden definir tomando en cuenta las máximas velocidades existentes en la actualidad. El conjunto B está en el intervalo de 0 a 1000, el conjunto M está de 500 a 1500 y el conjunto A se ubica entre los 1000 y 2000 MHz y por último el conjunto MA está definido para dar soporte a velocidades superiores a 2000 MHz. Dichos velocidades pueden ser modificadas en la función fuzzificación, la cual es explicada en la subsección 6.2.2.

Gráfica difusa del espacio disponible del disco duro (MB)

Para evaluar el espacio disponible en el disco duro, el universo de discurso está formado por los conjuntos `Bajo (B)`, `Medio (M)` y `Alto (A)`, el espacio disponible está definido en MB de 0 a 8000. Para un espacio disponible más grande se ha definido el conjunto `Muy Alto (MA)`, el cual da soporte a más de 8000 MHz, pero como solo se puede procesar una de tres posibles soluciones en las reglas de inferencia (B, M yA) la solución arrojada sería el conjunto **A**. En la figura 6.7 se representa el enfoque difuso de como se evaluá el espacio disponible en el disco duro.

Los intervalos definidos en el espacio disponible de la gráfica 6.7 son relativos tomando en cuenta el espacio disponible del disco duro en el momento de la petición. Como se puede observar el conjunto B se encuentra en el intervalo de 0 a 4000, el conjunto M está de 2000 a 6000 y el conjunto A se ubica entre los 4000 a 8000 MB y finalmente el conjunto MA está definido para dar soporte a un espacio disponible

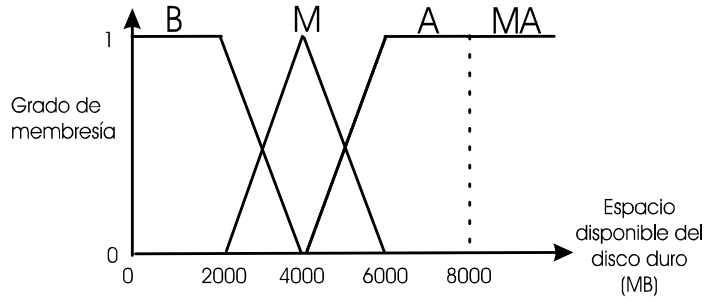

Figura 6.7: Representación gráfica difusa para el disco duro

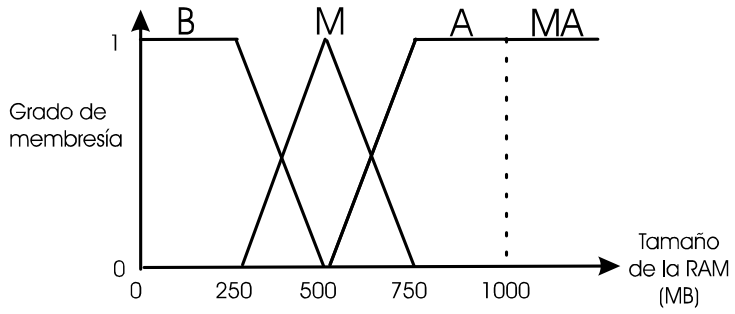

Figura 6.8: Representación gráfica difusa para la memoria principal

superior a 8000 MB.

Gráfica difusa del tamaño de la RAM (MB)

Para la evaluación del tamaño de la RAM, el universo de discurso está formado por los conjuntos `Bajo (B)`, `Medio (M)` y `Alto (A)`, el tamaño es medido en MB de 0 a 1000. Para tamaños de memoria más altos se tiene definido el conjunto `Muy Alto (MA)`, el cual da soporte a más de 1000 MB, pero como solo se puede procesar una de tres posibles soluciones en las reglas de inferencia (B, M y A) la solución obtenida será el conjunto **A**. En la figura 6.8 se representa el enfoque difuso de cómo se evaluá el tamaño de la memoria RAM.

Los intervalos de tamaños de memoria definidos en la gráfica 6.8 son relativos a los que se pueden definir tomando en cuenta actualmente los tamaños máximos disponibles. El conjunto B está en el intervalo de 0 a 500, el conjunto M está de 250 a 750 y el conjunto A se ubica entre los 500 y 1000 MB y por último el conjunto MA está definido para dar soporte a velocidades superiores a 1000 MB.

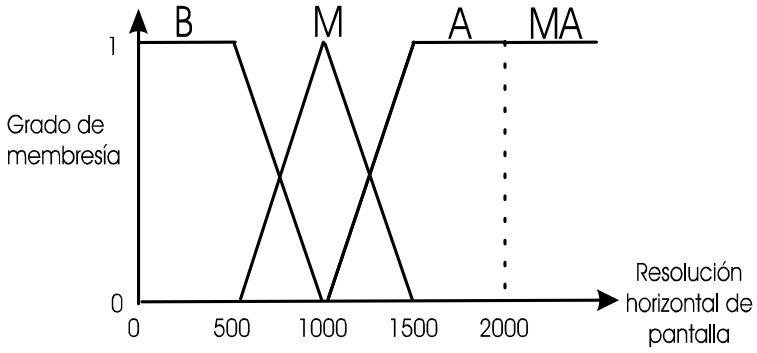

Figura 6.9: Representación gráfica difusa para la resolución de pantalla

Gráfica difusa de la resolución horizontal de pantalla

En el objetivo de evaluar la resolución horizontal de pantalla, el universo de discurso está formado por los conjuntos `Bajo (B)`, `Medio (M)` y `Alto (A)`, en el rango de 0 a 2000 pixeles. Para resoluciones horizontales altas se ha definido el conjunto `Muy Alto (MA)`, el cual da soporte a más de 2000 pixeles, pero como solo se pueden procesar una de tres posibles soluciones en las reglas de inferencia (B, M y A) la solución obtenida es el conjunto **A**. En la figura 6.9 se representa el enfoque difuso de cómo se evaluá la resolución horizontal de pantalla.

Los intervalos de resoluciones horizontales definidas en la gráfica 6.9 son relativas a las que se pueden definir considerando las actuales resoluciones. El conjunto B está en el intervalo de 0 a 1000, el conjunto M está de 500 a 1500 y el conjunto A se ubica entre los 1000 a 2000 MHz y el conjunto MA está definido para dar soporte a resoluciones horizontales superiores a 2000 MHz.

Gráfica difusa del tamaño de la entidad compartida (KByte)

Para evaluar el tamaño de cada entidad compartida, el universo de discurso está formado por los conjuntos `Bajo (B)`, `Medio (M)` y `Alto (A)`, dicho tamaño es medido en KByte de 0 a 5000. En el caso de que el tamaño de la entidad sea muy alto, se ha definido el conjunto `Muy Alto (MA)`, el cual da soporte a más de 5000 KByte, pero como solo se pueden procesar una de tres posibles soluciones en las reglas de inferencia (B, M y A) la solución arrojada es el conjunto **A**. En la figura 6.10 se representa el enfoque difuso de cómo se evalua el tamaño de la entidad compartida.

Los intervalos de los tamaños de las entidades definidos en la gráfica 6.10 son relativos a los que se pueden definir tomando en cuenta que una entidad compartida puede llegar a ser muy grande. El conjunto B está en el intervalo de 0 a 2500, el conjunto M está de 1250 a 3750 y el conjunto A se ubica entre los 2500 a 5000 KB y por último el conjunto MA está definido para dar soporte a tamaños superiores a 5000 KB.

Una vez expuestos los rangos de los conjuntos difusos de cada variable lingüística

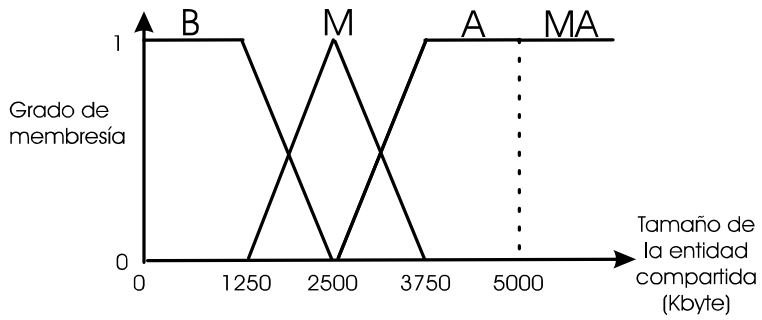

Figura 6.10: Representación gráfica difusa para el tamaño de la entidad compartida

se procede a explicar el método de la `fuzzificación`, cuyo objetivo es determinar el grado de membresia de cada entrada precisa de las variables lingüísticas y así obtener los conjuntos difusos representativos (B, M o A) de las entidades evaluadas para utilizarse en el motor de inferencia y así definir las versiones más adecuadas para enviarlas al sitio participante.

6.2.2. Fuzzificación

Para el desarrollo difuso de la arquitectura de distribución adaptable se utilizaron las `funciones de pertenencia triangular y trapezoidal` (ver subsección 3.1.2). Los `términos` utilizados son: B (Bajo), M (Medio), A (Alto) y MA (Muy Alto), las `variables lingüísticas` son la velocidad del procesador en MHz, el espacio disponible del disco duro en MB, tamaño de la memoria primaria en MB, la resolución de la pantalla en base al número de pixeles horizontales y al tamaño de la entidad evaluada en KByte. El `universo de discurso` está integrado por los límites de los conjuntos difusos, los cuales están definidos para cada una de las variables lingüísticas,

Los conjuntos difusos se dividen en una parte baja y una alta. De esta forma se obtienen seis subconjuntos difusos cuya estructura está dada por la forma `conjunto-subconjunto`, por lo tanto las agrupaciones obtenidas son: bajo-baja y bajo-alta, medio-baja y medio-alta, alto-baja y alto-alta (ver figura 6.11).

Parámetros de entrada

Los límites mostrados en la figura 6.11 son:

- **LPSD**. El **Límite del Primer Subconjunto Difuso** indica donde está el mayor grado de membresía del conjunto B (Bajo), además es la división entre las partes baja y alta de dicho conjunto. También hace referencia al menor grado de membresía de subconjunto medio-baja.
- **LSTSD**. El **Límite del Segundo y Tercer Subconjunto Difuso** especifica donde está el mayor grado de membresía del conjunto M (Medio) y la división

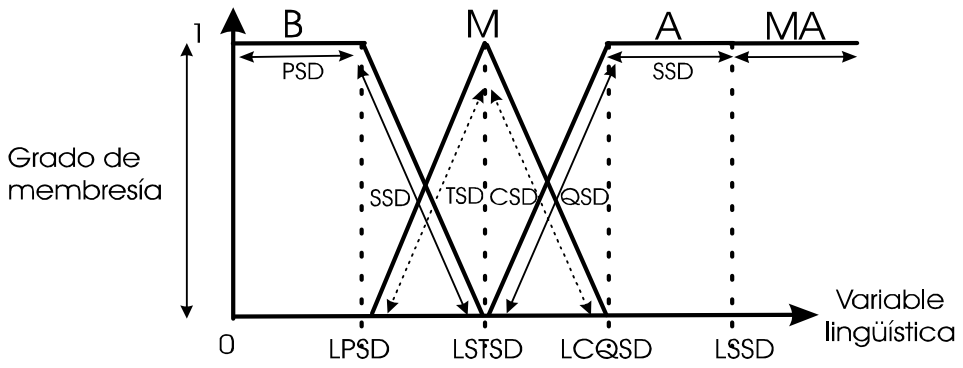

Figura 6.11: Límites de los subconjuntos difusos

de sus partes baja y alta. Además es el punto donde recide el menor grado de membresía de los subconjuntos bajo-alta y alto-baja.

- **LCQSD**. El **Límite del Cuarto y Quinto Subconjunto Difuso** indica donde está el mayor grado de membresía del conjunto A (Alto), la división de sus partes baja y alta y además el menor grado de membresía del subconjunto medio-alta.

- **LSSD.** El **Límite del Sexto Subconjunto Difuso** representa la consecución del conjunto A y sirve para soportar *entradas precisas* cuyo valor exceda al LSSD. Una **entrada precisa** corresponde al valor real de la entidad compartida evaluada en un momento determinado y la cual forma parte de alguna variable lingüística.

Conjuntos y subconjuntos difusos

Para determinar cual es el conjunto difuso más adecuado en base al valor de una entrada precisa de una variable lingüística, se hace uso de las funciones de pertenencia triangular y trapezoidal y el universo de discurso es dividido en varios subconjuntos difusos. Como puede observarse en la figura 6.12(a, b, c, d, e y f) existen seis subconjuntos difusos bien definidos: PSD, SSD, TSD, CSD, QSD y SSD los cuales están de color gris. Existe un séptimo sub-conjunto llamado SepSD el cual forma parte del conjunto difuso MA. Dicho conjunto es tratado de tal forma que cualquier acción realizada en él y una vez obtenido el resultado se toma como parte del conjunto difuso A, esto se lleva acabo así por que las reglas de inferencia solo pueden recibir y procesar tres posibles valores (B. M y A) para cada una de las variables lingüísticas (velocidad del procesador, espacio disponible del disco duro, tamaño de la RAM y la resolución horizontal de la pantalla).

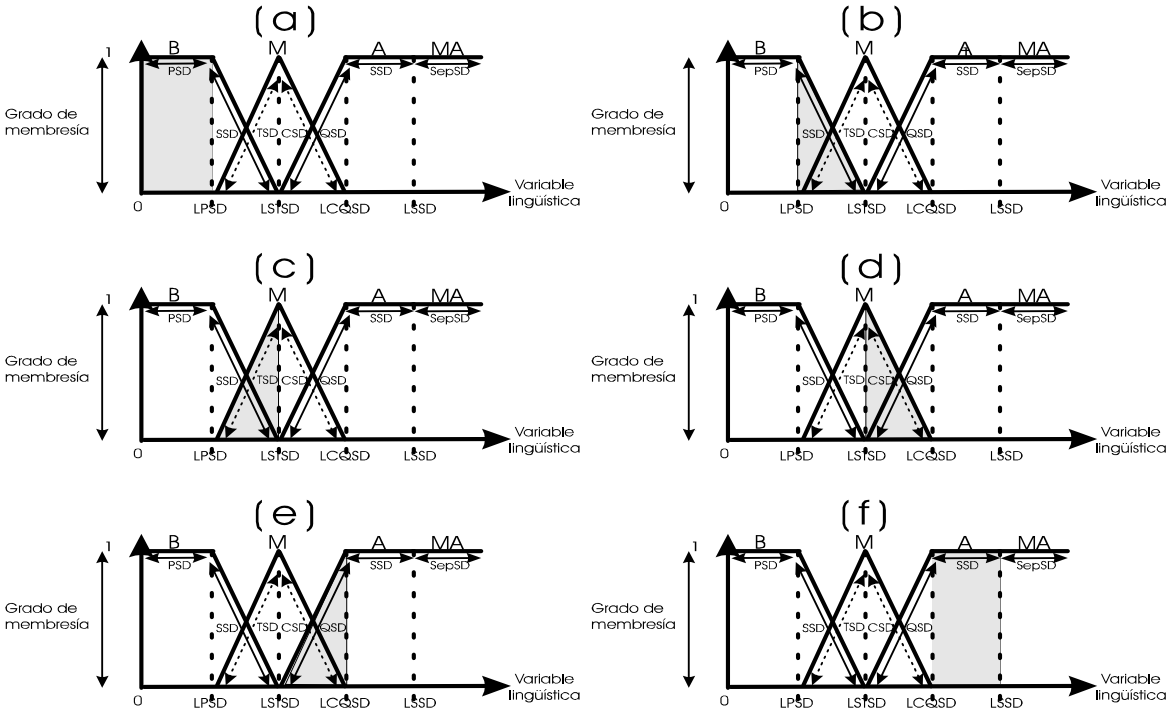

Figura 6.12: Límites de los subconjuntos difusos

Funciones de pertenencia triangular-trapezoidal

La representación matemática del proceso de fuzzificación está dado por siete funciones y sus respectivas condiciones en los límites de los subconjuntos difusos. Cada una de estas funciones corresponde a los subconjuntos difusos mostrados anteriormente.

Siendo $A = 1$ y EP (entrada precisa) corresponde al valor real de una variable lingüística.

- Primer Subconjunto Difuso (PSD), ver figura 6.12-a

$$\mu(EP) = A \qquad 0 < EP \leq LPSD$$

- Segundo Subconjunto Difuso (SSD), ver figura 6.12-b

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LPSD) \qquad LPSD < EP \leq LSTSD$$

- Tercer Subconjunto Difuso (TSD), ver figura 6.12-c

$$\mu(EP) = A(LPSD - EP)/(LPSD - LSTSD) \qquad LPSD < EP \leq LSTSD$$

- Cuarto Subconjunto Difuso (CSD), ver figura 6.12-d

$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \qquad LSTSD < EP \leq LCQSD$

- Quinto Subconjunto Difuso (QSD), ver figura 6.12-e

$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \qquad LSTSD < EP \leq LCQSD$

- Sexto Subconjunto Difuso (SSD), ver figura 6.12-f

$\mu(EP) = A \qquad LCQSD < EP \leq LSSD$

- Septimo Subconjunto Difuso (SepSD)

$\mu(EP) = A \qquad EP > LSSD$

Procedimiento de fuzzificación

La metodología para implementar el mecanismo de fuzzificación es el siguiente:

1. definir una función para contener todo el proceso de fuzzificación, la cual reciba los siguientes parámetros: EP, $LPSD$, $LSTSD$, $LCQSD$ y $LSSD$,
2. definir una variable entera la cual contendrá al conjunto difuso resultante,
3. definir un *if* inicial para la función **PSD**, la cual debe tener la condición $0 < EP \leq LPSD$ cuyo grado de pertenencia será 1 y la variable resultante será el conjunto difuso **Bajo**,
4. definir un *else-if* para las funciones **SSD** y **TSD** con la condición $LPSD < EP \leq LSTSD$,
5. dentro del anterior *else-if* definir un *if-else* en el que se compare y determine cual es el máximo grado de pertenencia entre los resultados de las funciones **SSD** y **TSD**. En caso de que **SSD** tuviera el máximo grado de pertenencia se le asignaría a la variable resultante el conjunto difuso **Bajo** y si fuera **TSD** se le asignaría **Medio**,
6. definir un *else-if* para las funciones **CSD** y **QSD** con la condición $LSTSD < EP \leq LCQSD$,
7. dentro del anterior *else-if* se define un *if-else* en el que se compare y determine cual es el máximo grado de pertenencia entre los resultados de las funciones **CSD** y **QSD**. En caso de que **CSD** tuviera el máximo grado de pertenencia se le asignaría a la variable resultante el conjunto difuso **Medio** y si fuera **QSD** se le asignaría **Alto**,
8. definir un *else-if* para la función **SSD**, la cual debe tener la condición $LCQSD < EP \leq LSSD$ cuyo grado de pertenencia será 1 y la variable resultante será el conjunto difuso **Alto**,

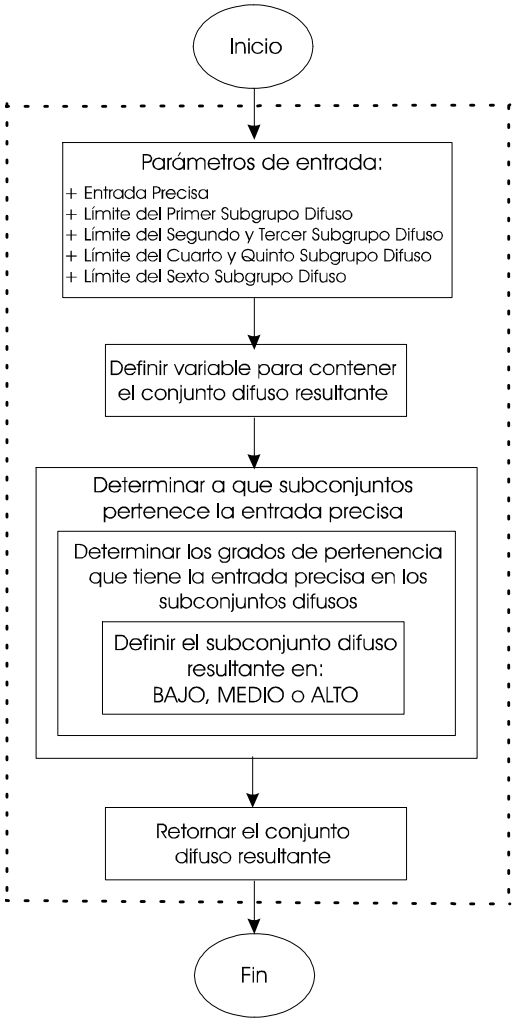

Figura 6.13: Procedimiento general de la fuzzificación

9. definir un *else-if* para la función **SepSD**, la cual debe tener la condición $EP > LSSD$ cuyo grado de pertenencia será 1 y la variable resultante será el conjunto difuso **Alto**,

10. el *else* debe tener la funcionalidad de abortar el sistema en caso de que la entrada precisa tenga un valor menor o igual a 0

11. retornar el conjunto difuso resultante

En la figura 6.13 se muestran las partes que integran el módulo de fuzzificación.

6.2.3. Motor y base de reglas de inferencia

Entradas al motor de inferencia

Como se puede observar el motor de inferencia y la base de reglas de inferencia constituyen dos componentes que están combinados. Este módulo recibe seis parámetros representados por las variables: X1, X2, X3, X4, X5 y Categoría. Los primeros cuatro pertenecen a los valores difusos correspondientes a las capacidades de los dispositivos arrojados por el módulo de fuzzificación. El quinto parámetro corresponde al valor difuso del tamaño de la entidad compartida y el último es el tipo de categoría a la que pertenece dicha entidad compartida. La estructura general del motor de inferencia y la base de reglas de inferencia se muestra en la figura 6.14.

Categorias del motor de inferencia

Para poder agilizar la búsqueda en la base de reglas de inferencia, esta ha sido dividida en cinco categorías correspondientes a cinco tipos de entidades multimedia: texto, imagen, animación, audio y vídeo. El total de reglas es de 1215 distribuidas en cinco grupos de 243 reglas.

El número de reglas de inferencia por categoría es determinado mediante la siguiente ecuación: $a \cdot a \cdot a \cdot a \cdot a$, donde $a = 3$ y 3 simboliza a los conjuntos difusos (B, M y A), Toda la ecuación representa respectivamente a las cuatro capacidades del sitio participante (velocidad del procesador, espacio disponible en el disco duro, tamaño de la RAM, la resolución horizontal de la pantalla) y por último al tamaño de la entidad compartida. Dicha ecuación da un total de 243 reglas, las cuales son multiplicadas por las cinco categorías multimedia, dando un total de 1215 reglas de inferencia.

La definición de cinco categorías de reglas para evaluar las entidades compartidas se justifica para evitar que se realice una búsqueda por las 1215 reglas de inferencia al momento de evaluar alguna entidad. De esta forma el máximo de reglas analizadas será de 243. El archivo `mime.types` se toma como referencia para poder establecer un listado con las extensiones que soporta el servidor Web Apache, dicho listado es utilizado para determinar a que categoría corresponde la entidad compartida que se está evaluando.

Estructura de las reglas de inferencia

Para mostrar como están estructuradas las reglas de inferencia se presenta la primera y última regla de cada una de las categorías:

```
if(categoria == esTexto){
if((X1 == B)&&(X2 == B)&&(X3 == B)&&(X4 == B)&&(X5 == B))
    {
        printf("Rule1\n");
        returnR3;
    }
```

```
    ...
if((X1 == A)&&(X2 == A)&&(X3 == A)&&(X4 == A)&&(X5 == A))
    {
        printf("Rulea243\n");
        returnR3;
    }
}
if(categoria == esImagen){
if((X1 == B)&&(X2 == B)&&(X3 == B)&&(X4 == B)&&(X5 == B))
    {
        printf("Rule244\n");
        returnR3;
    }
    ...
if((X1 == A)&&(X2 == A)&&(X3 == A)&&(X4 == A)&&(X5 == A))
    {
        printf("Rule486\n");
        returnR3;
    }
}
if(categoria == esAnimacion){
if((X1 == B)&&(X2 == B)&&(X3 == B)&&(X4 == B)&&(X5 == B))
    {
        printf("Rule487\n");
        returnR1;
    }
    ...
if((X1 == A)&&(X2 == A)&&(X3 == A)&&(X4 == A)&&(X5 == A))
    {
        printf("Rule729\n");
        returnR3;
    }
}
if(categoria == esAudio){
if((X1 == B)&&(X2 == B)&&(X3 == B)&&(X4 == B)&&(X5 == B))
    {
        printf("Rule730\n");
        returnR1;
    }
    ...
if((X1 == A)&&(X2 == A)&&(X3 == A)&&(X4 == A)&&(X5 == A))
    {
        printf("Rule972\n");
```

```
        returnR3;
    }
}
if(categoria == esVideo){
if((X1 == B)&&(X2 == B)&&(X3 == B)&&(X4 == B)&&(X5 == B))
    {
        printf("Rule973\n");
        returnR1;
    }
    ...
if((X1 == A)&&(X2 == A)&&(X3 == A)&&(X4 == A)&&(X5 == A))
    {
        printf("Rule1215\n");
        returnR3;
    }
}
```

Como se observa en las reglas de inferencia, se hace uso de cinco *if* para establecer las categorias multimedia. Posteriormente cada *if* incluye bloques *ifs* anidados, uno para cada regla definida. Las reglas difusas están estructuradas por las siguientes partes:

1. los `parámetros` que contienen el término difuso (B, M y A) para cada una de las cuatro capacidades evaluadas del sitio participante (X1, X2, X3 Y X4) y del tamaño de la entidad compartida (X5),

2. los `términos difusos` con los que se comparan y evaluán los parámetros: B = 1 corresponde al conjunto difuso Bajo, M = 2 al conjunto difuso Medio y A = 3 al conjunto difuso Alto,

3. los `resultados arrojados` por las reglas de inferencia: V1 = 1 es la versión de la entidad compartida con una mayor degradación, V2 = 2 es la versión de la entidad compartida con una degradación menor a V1, V3 = 3 es la versión que tiene menor degradación o ninguna y S = 0 es el substituto.

En la figura 6.14 se muestra de una forma general como está estructurado el motor y la base de reglas de inferencia.

6.2.4. Cargar el listado de las entidades aceptadas

Inicialmente se definen dos estructuras, una llamada `entidades_elegidas` y la otra llamada `temporal` y los campos que las integran son: `idEntidad`, `nombreEntidad`, `derechoAcceso`, `rutaEntidad`, `tamArchivo` y `aceptadoRechazado`. Una vez definidas las estructuras se abre un flujo al archivo `listadoEntidadesElegidas.lst`, el cual y como su nombre lo dice, contiene los registros de las entidades compartidas que

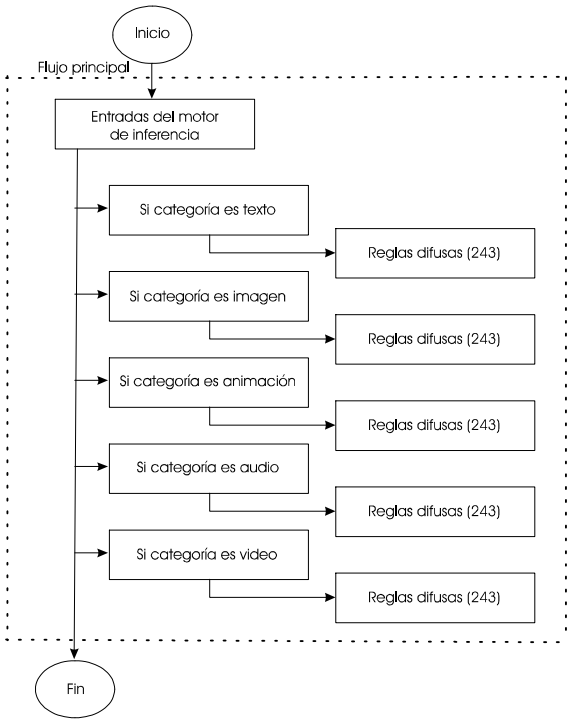

Figura 6.14: Motor de inferencia

el usuario ha seleccionado para recibir en su dispositivo. Posteriormente se cargan los datos a la estructura `entidades_elegidas` y los registros que en su campo `aceptadoRechazado` contienen una A son filtrados a la estructura `temporal`, lo cual significa que el recurso compartido ha sido aceptado por el usuario (ver figura 6.15).

6.2.5. Entidades soportadas por el software existente en el sitio participante

En este apartado se determina si una entidad compartida es soportada o no por el software existente en el sitio participante. Lo anterior se realiza mediante el uso de las extensiones de las entidades, las cuales se guardan en un archivo llamado `extensiones.lst` (ver figura 6.16), el cual está contenido en el sitio de almacenamiento. La implementación de este módulo se divide en dos partes:

- la primera es la obtención de las extensiones de cada una de las categorias y
- la segunda corresponde en comparar y determinar a que categoría pertenece la entidad evaluada (ver figura 6.17).

En la figura 6.16 se muestra el formato del contenido en el archivo `extensiones.lst`. Como puede observarse, existen cinco listados, cada uno es referente a un `tipo/subtipo`

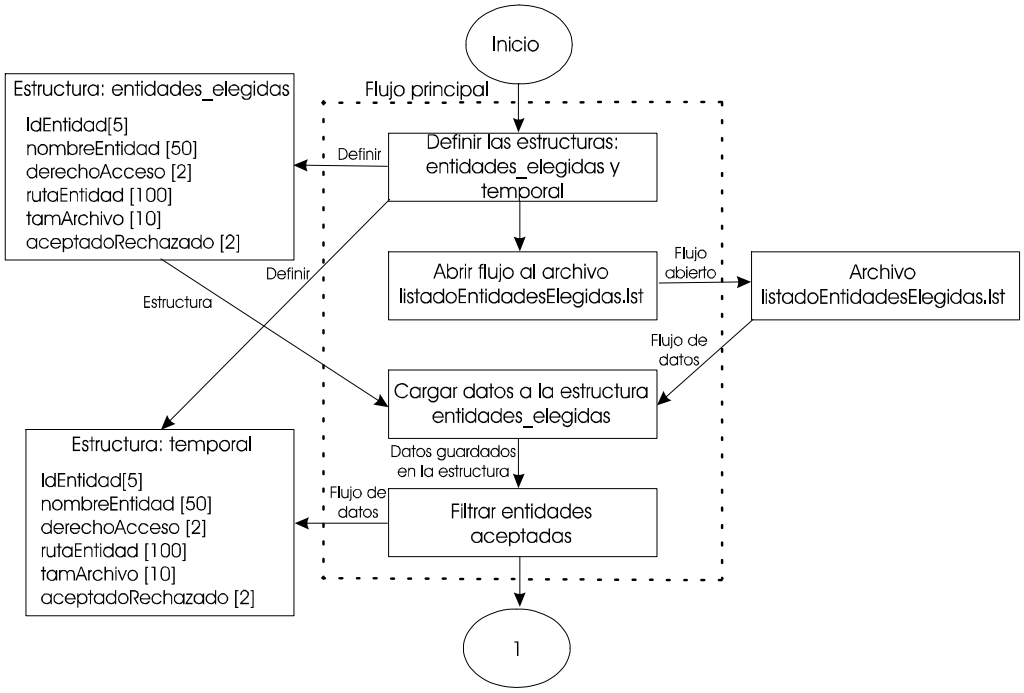

Figura 6.15: Cargar el listado de las entidades aceptadas

`mime.types` manejado en el sistema, dichos listados son de: texto, imagen, animación, audio y vídeo. Las etiquetas que identifican a las categorias son: —TEXT—, —IMAGE—, —ANIMATION—, —AUDIO— Y —VIDEO— y sus correspondientes extensiones se definen a continuación.

Para cada sitio participante que se conecte al sitio de almacenamiento se debe generar un archivo con este formato, ya que con este archivo se están definiendo las extensiones que soporta el software instalado en el sitio participante. Estos archivos deben estar contenidos en el sitio de almacenamiento, para que así cuando un sitio participante haga una petición, entonces sea buscado el archivo `extensiones.lst` correspondiente. El sistema actualmente no tiene implementada la función de multiples archivos, así que solo tiene las extensiones para un sitio participante a la vez.

La carga de las extensiones mime.types al sistema de adaptación a dispositivos es mostrada en la figura 6.17-a

- Inicialmente se abre un flujo hacia el archivo `extensiones.lst` y se determina el total de extensiones existentes.
- Se define el arreglo `extensiones` cuya dimensión "`total_extensiones`" fue determinado automáticamente cuando se abrió el flujo hacia el archivo extensiones.lst.
- Se definen las `etiquetas tipo/subtipo mime.types` en una arreglo de caracteres, las cuales tienen la forma siguiente: *TEXT = –TEXT—", *IMAGE

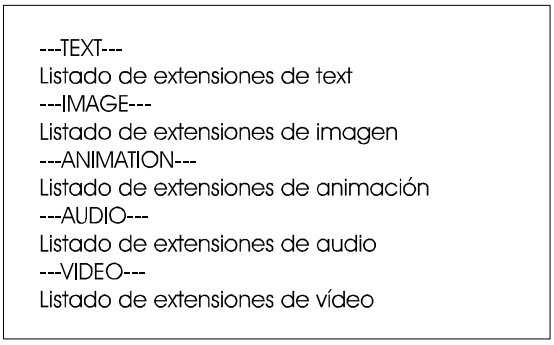

Figura 6.16: Formato del contenido del archivo de las extensiones

= –IMAGE—", *ANIMATION = –ANIMATION—", *AUDIO = –AUDIO—", *VIDEO = –VIDEO—".

- A continuación se cargan las extensiones del archivo `extensiones.lst` y cada una se va comparando con las `etiquetas mime.types` anteriormente definidas. En el caso de ser iguales se establece una variable indicando la `posición de la etiqueta` y a que tipo mime.types pertenece. En caso de que la extensión sea diferente de la etiqueta, dicha extensión será guardada en el `arreglo extensiones`. La variable que contiene la posición de la etiqueta mime.types es para memorizar la localidad en que el arreglo extensiones inicia el listado de un determinado tipo mime.types de extensiones y además sirve como límite para el listado anterior, por tal motivo deben haber cinco variables, e.g., limite_texto, limite_imagen, limite_animacion, limite_audio y limite_video.

El procedimiento para determinar si la entidad evaluada es soportada por el software existente en el sitio participante es mostrado en la figura 6.17-b.

- Como se observa en la parte media de la figura 6.17 se describe el inicio de un ciclo, el cual tiene la función que por cada pasada o iteracción se evaluá una de los entidades contenidas en el listado de entidades compartidas que han sido aceptados por el usuario. También el contenido mostrado en las figuras 6.19 y 6.20 forman parte de este ciclo.

- En la etapa siguiente, se obtiene la extensión de la entidad evaluada, la cual esta contenida en el campo `nombreEntidad` de la estructura temporal.

- Una vez obtenida dicha extensión es comparada con el contenido del `arreglo extensiones`, si existe alguna correspondencia se continua con el siguiente paso del procedimiento, en caso contrario la entidad compartida no es soportado y se termina la evaluación de este mostrando el mensaje adecuado a esta situación y se inicia la evaluación de la siguiente entidad de la lista.

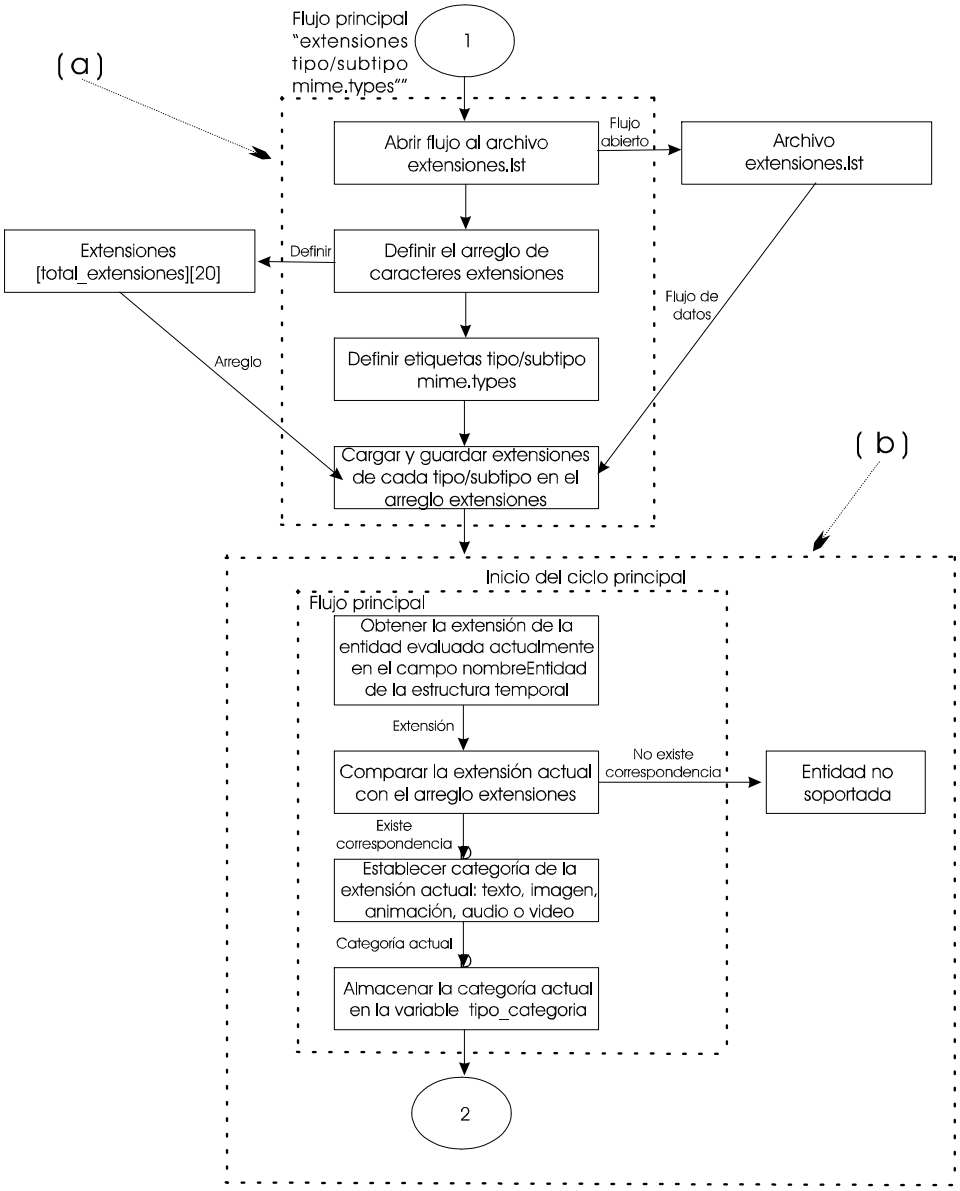

Figura 6.17: Procedimiento para la definición del sotfware existente y de las categorías

```
if (indice >= limite_texto && indice < limite_imagen)
  tipo_categoria = esTexto;
else if (indice >= limite_imagen && indice < limite_animacion)
  tipo_categoria = esImagen;
else if (indice >= limite_animacion && indice < limite_audio)
  tipo_categoria = esAnimacion;
else if (indice >= limite_audio && indice < limite_video)
  tipo_categoria = esAudio;
else if (indice >= limite_video && indice < ind_ext_ext)
  tipo_categoria = esVideo;
else
  printf ("No existe la categoría\n");
```

Figura 6.18: Definición de las categorías

- El siguiente paso es establecer la categoría a la que corresponde la entidad compartida en base a su extensión. Para llevar acabo dicho procedimiento se hace uso de las `variables de los límites` definidas anteriormente, ya que con estas se da un intervalo de las extensiones que corresponden a cada tipo/subtipo mime.types. Se toma el `índice de la correspondencia` encontrada (posición de la extensión en el arreglo extensiones) en el paso anterior para compararse con los límites y así determinar a que categoría pertenece la entidad. Por último el resultado es guardado en la variable `tipo_categoría` (ver figura 6.18).

6.2.6. Cargar capacidades del dispositivo

Inicialmente se define una estructura llamada `capacidades`, cuyos campos que la integran son: `procesador`, `discoDuro`, `RAM` y `resolPantalla`. Posteriormente se abre un flujo al archivo `listadoCapacidades.lst`, el cual contiene los registros de las capacidades del sitio participante. A continuación dichos registros son cargados a la estructura capacidades y una vez ahí son convertidos a tipo entero para poder pasarlos al módulo de fuzzificación (ver figura 6.19). Las capacidades son cargadas solo una vez al sistema para la evaluación de cada una de las entidades compartidas solicitadas por un usuario.

Otro dato que es utilizado en esta sección es el `tamaño de la entidad` compartida, el cual es obtenido del campo `tamArchivo` de la estructura temporal (ver figura 6.15). De igual forma que los datos de las capacidades son convertidos a tipo entero, también el tamaño de la entidad.

6.2.7. Llamada a la fuzzificación y al motor de inferencia

Finalmente, en esta sección se presenta el ciclo principal de la evaluación de la entidad compartida. Este ciclo usa y llama a la `función fuzzificacion`, la cual

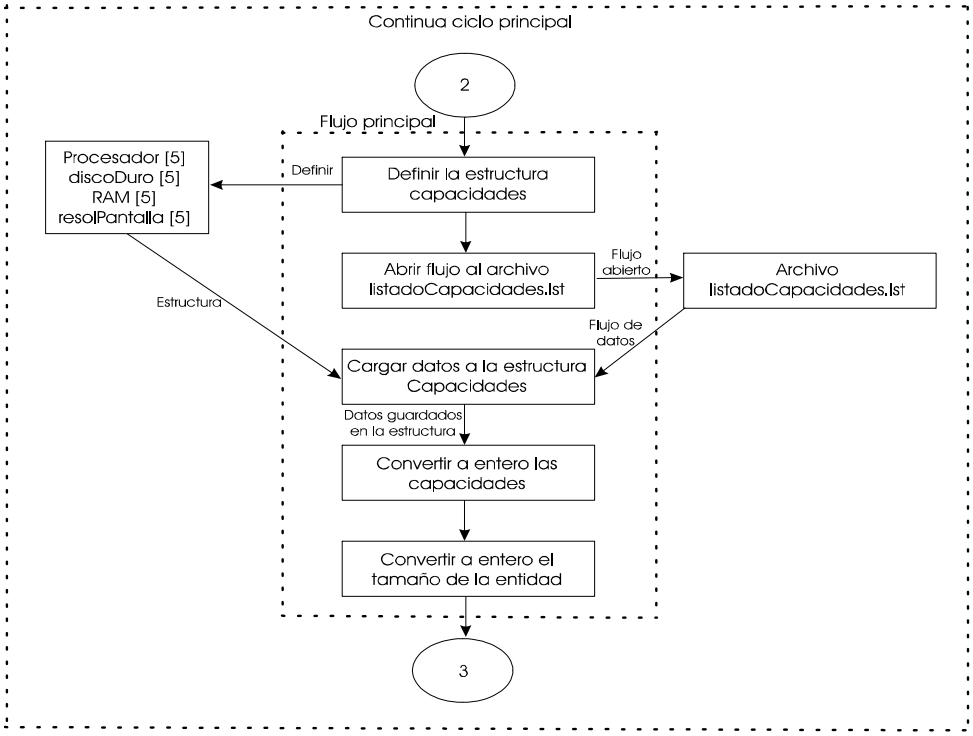

Figura 6.19: Recuperación de las capacidades del dispositivo

recibe como parámetros:

- la entrada precisa (EP) o el dato a evaluar (cada una de las capacidades del sitio participante y el tamaño de la entidad),
- el límite del primer subconjunto difuso (LPSD),
- el límite del segundo y tercer subconjunto difuso (LSTSD),
- el límite del cuarto y quinto subconjunto difuso (LCQSD) y
- el límite del sexto subgrupo difuso (LSSD) (ver subsección 6.2.2).

La función de fuzzificación es llamada cinco veces:

- una para evaluar la velocidad del procesador,
- otra para el espacio disponible en el disco duro,
- otra para el tamaño de la memoria RAM,
- otra para la resolución horizontal de la pantalla y
- la última para el tamaño de la entidad compartida.

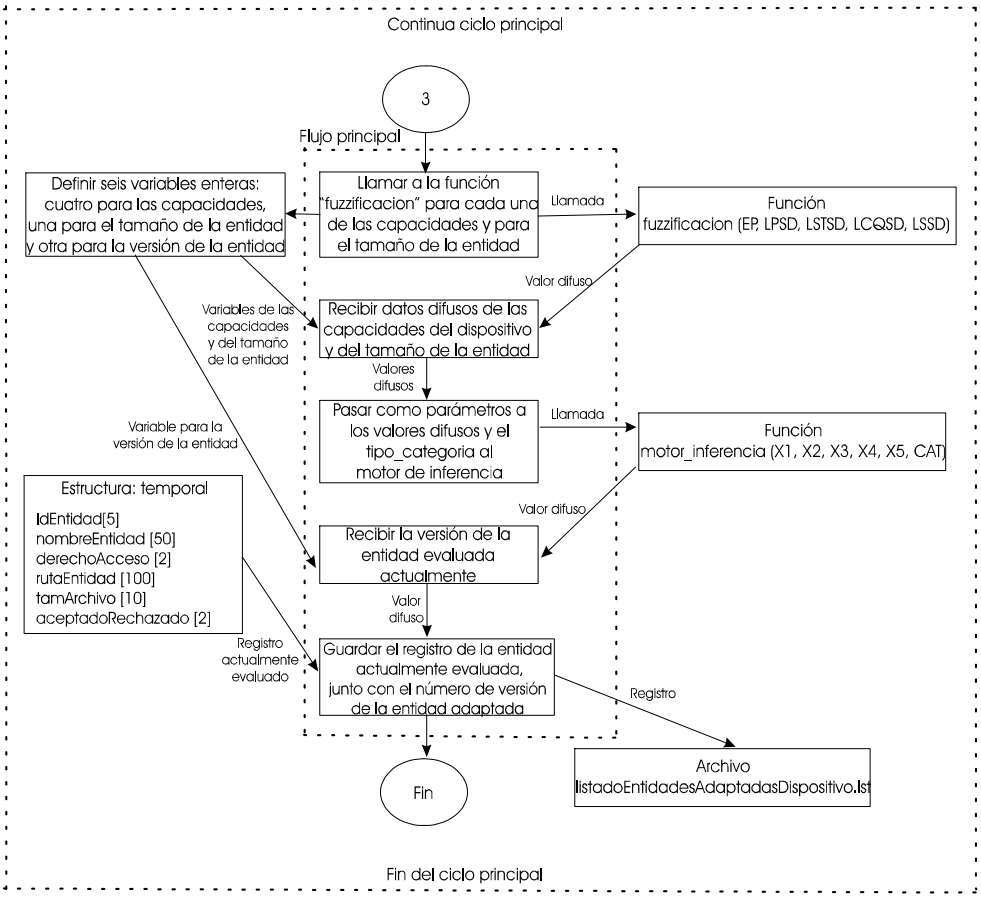

Figura 6.20: Llamada a las funciones de fuzzificación y al motor de inferencia

Los cinco valores difusos obtenidos son pasados como parámetros a la función `motor_inferencia` (ver subsección 6.2.3). Existe un sexto parámetro, el cual hace referencia a la categoría que pertenece la entidad evaluada. El resultado obtenido es la `versión adecuada al sitio participante de la entidad compartida` evaluada `en este ciclo`. Por último, el registro de la entidad evaluada y el número de la versión adecuada son guardados en el archivo listadoEntidadesAdaptadasDispositivo.lst (ver figura 6.20).

6.3. Mecanismo de distribución de las entidades

En esta sección se explica de forma general la funcionalidad del mecanismo de distribución de entidades compartidas. Posteriormente se exponen las ampliaciones realizadas para dar soporte a la distribución de las entidades compartidas adaptadas entre los sitios participantes y los sitios de almacenamiento. Dichas ampliaciones tienen un enfoque peer to peer, el cual es representado de dos formas: 1) en la red

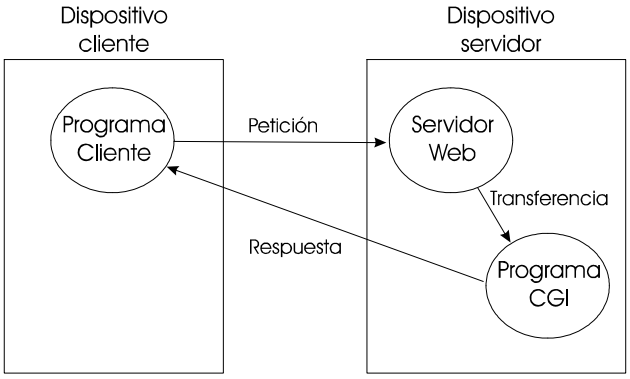

Figura 6.21: Funcionamiento de una llamada script CGI

local, la cual está formada por un sitio de almacenamiento y varios sitios participantes y 2) la red extendida, en la cual solo existen sitios de almacenamiento.

6.3.1. Base para la definición del mecanismo de distribución

Para realizar la comunicación entre los sitios participantes se utiliza como base la interfaz de comunicación de la plataforma PIÑAS, a la cual se le agregó funcionalidad para darle un enfoque peer to peer, porque que originalmente la interfaz tiene un enfoque cliente-servidor. La interfaz está integrada por un conjunto de *scripts* CGI definidos en el lenguaje C, dichos *scripts* utilizan e interaccionan con el protocolo HTTP del servidor Apache. La interfaz está empaquetada en el directorio /CGIappli. Los archivos fuente modificados son: `essaipost.c`, `pscript.c` y `wwwEssai.c`:

essaipost.c es el programa cliente, el cual al ejecutarse solicita el *host*, el puerto y el nombre de la entidad a transmitir. Una vez proporcionada dicha información se pasan los parámetros a wwwEssai, el cual obtiene la ruta destino, la longitud de la entidad a transferir y posteriormente estos datos son enviados al *script* pscript. Cuando pscript se ejecuta y envía los resultados a essaipost, este último los recibe en forma de bloques y los guarda en un archivo llamado *back.out*.

pscript.c reside en el directorio cgi-bin del servidor Apache, desempeña la tarea de retornar los bloques de información que integrarán el archivo *back.out*.

wwwEssai.c recibe los parámetros que son enviados por essaipost para que se determine el nombre y dirección IP del sitio participante y así pscript realice la transferencia de la entidad. El *script* wwwEssai es el punto de conexión entre essaipost y pscript.

El funcionamiento general de una llamada *script* CGI se puede observar en la figura 6.21

6.3.2. Funcionamiento global del mecanismo de distribución

El mecanismo de distribución de entidades compartidas adaptadas está especificado en dos partes: la primera corresponde a la transmisión de entidades y la segunda a la recepción de estas entidades. Dichas partes están implementadas en el *script* CGI denominado **pscript**. En cierta forma este *script* es la unión del pscript.c inicial y de essaipost.c, más algunas condiciones para que se determine en que momento el mecanismo de distribución actúa como cliente o como servidor.

Mecanismo de transmisión

El mecanismo de transmisión es el encargado de enviar las entidades al sitio participante que las haya solicitado. El funcionamiento comienza al ejecutar el comando **ls -1**, cuyo resultado se redirecciona al archivo `ListadoArchivos`. Posteriormente se define la estructura `archivos_directorio` con los campos `IdArchivo [5]` y `nombreArchivo [81]`. Se abre un flujo al archivo `ListadoArchivos` y se cargan los datos a la estructura `archivos_directorio`. Lo anterior sirve para obtener los nombres de los archivos de `autor-multisitios`, los cuales se encuentran en un directorio específico.

Posteriormente se define la estructura `fichaPINAS` con los campos *host* y `puerto`. Se establece un ciclo para obtener el *host* y `puerto` de todos los sitios participantes, para ello dentro del ciclo se abre un flujo al archivo llamado conforme al `registro N` de la estructura `archivos_directorio` y se cargan los datos a `fichaPINAS` **(ver** figura 6.22**)**. El *host* y puerto serán utilizados en la segunda parte del mecanismo de transmisión.

En la figura 6.23 se muestra la segunda parte del mecanismo de transmisión. Se establece un ciclo con el número máximo de las entidades a transferir para el usuario y por cada iteración se evaluá y transmite una entidad. La funcionalidad contenida en dicho ciclo inicia cuando se carga el listado de las entidades adaptadas al dispositivo (`listadoEntidadesAdaptadasDispositivo.lst`) a la estructura `entidadesAdaptadas`. Posteriormente se define la ruta del `contenedor origen` y la `ruta del contenedor destino remoto`, para ello se debe concatenar la `ruta del contenedor origen` con el `nombre de la entidad` a enviar. También se debe realizar la misma operación con la `ruta del contenedor destino remoto` y con el `nombre de la entidad` a enviar. Ahora bien, teniendo las dos rutas y tomando los datos del *host* y puerto contenidos en la estructura `fichaPINAS` se procede a pasar dichos datos como parámetros a la función `wwwEssai`. La función `wwwEssai` tiene los siguientes parámetros:

- *host*: nombre del sitio participante,
- *port*: número del puerto en el sitio participante,
- URL_origin: ruta del contenedor origen,

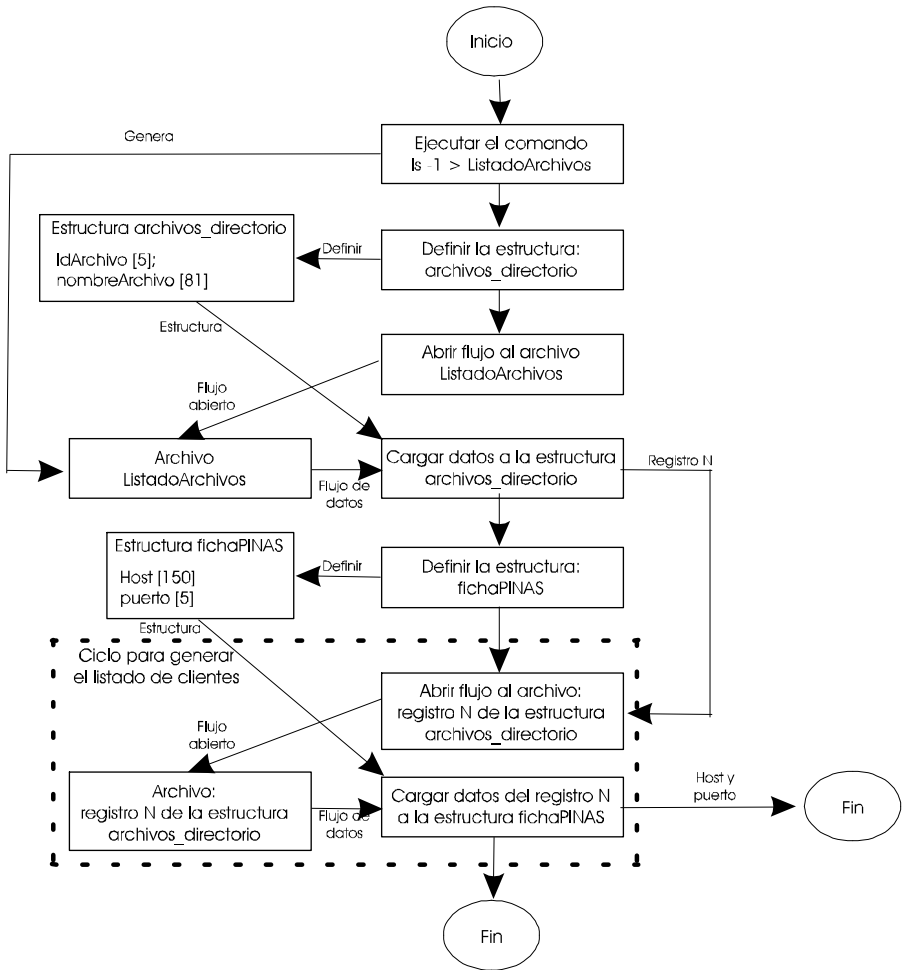

Figura 6.22: Mecanismo de transmisión de entidades compartidas adaptadas (1)

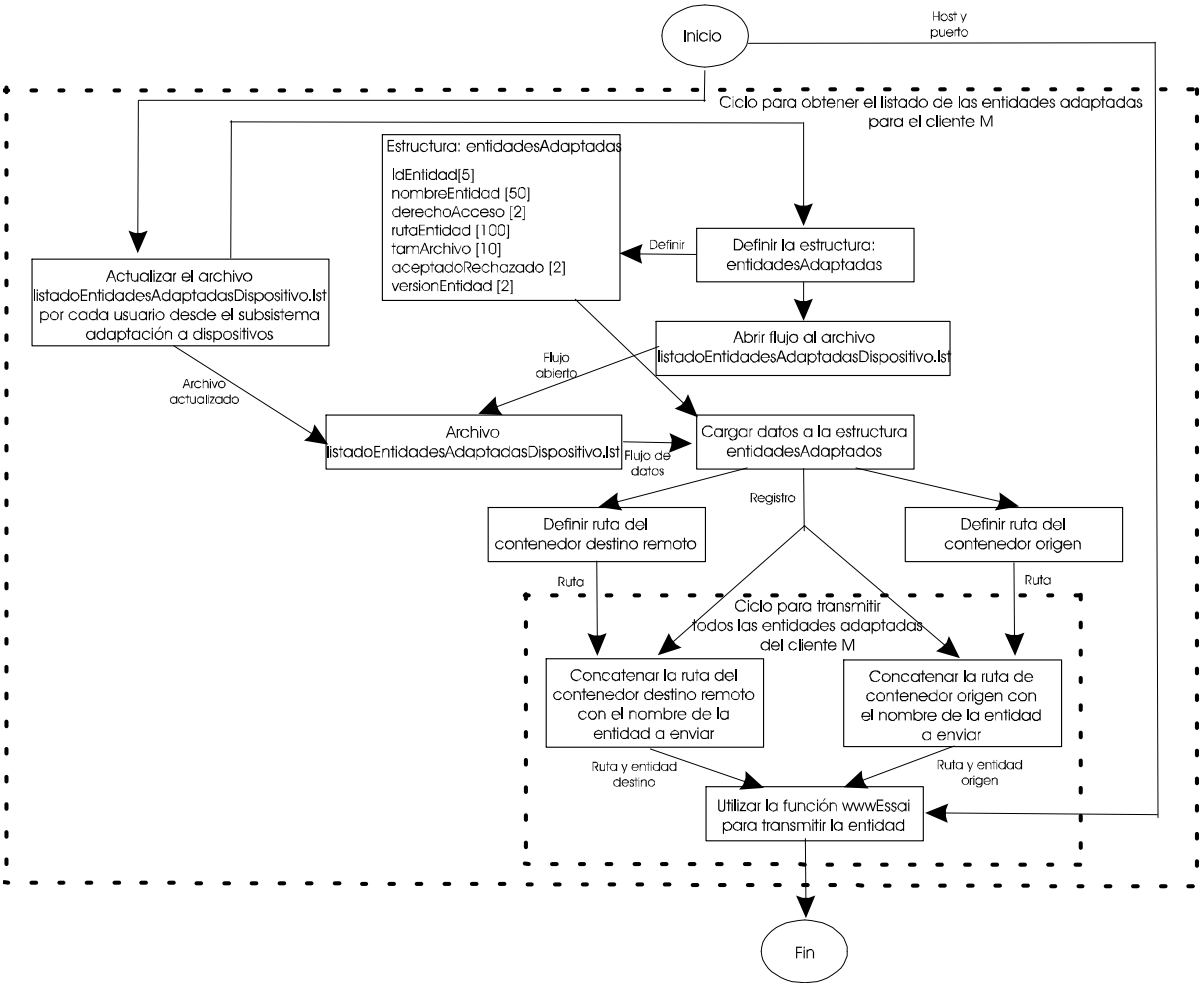

Figura 6.23: Mecanismo de transmisión de entidades compartidas adaptadas (2)

- *parameters*: es un arreglo que contiene la ruta destino remota y el nombre de la entidad adaptada a transmitir,
- &memAddr: buffer intermedio que contiene la entidad transmitida,
- &memLen: longitud en bytes del tamaño de la entidad compartida.

El proceso anterior sirve para la transmisión de una entidad compartida adaptada al sitio participante solicitante.

Mecanismo de recepción

El mecanismo de recepción se encarga de recibir las entidades compartidas adaptadas y guardarlas en el sitio de trabajo. Como se mencionó al inicio de la sección el

mecanismo de distribución utiliza un enfoque peer to peer el cual es representado en dos formas: en la red local y en la red extendida.

El módulo de red local se refiere a que en un ámbito local de redes de computadoras existe un sitio de almacenamiento y varios sitios de trabajo. En la figura 6.24 se muestra la funcionalidad de la recepción y guardado de una entidad entre un sitio de almacenamiento y un sitio de trabajo. Dicho procedimiento es compuesto de los siguientes pasos:

- Se utiliza la función `initScriptEnvironment`, la cual usa los siguientes parámetros: 1) `&paramNbr` que contiene el número de parámetros, 2) `params` contiene el nombre de los parámetros, 3) `&data` contiene el valor de la entidad compartida transmitida y 4) `&dataLen` contiene la longitud de la entidad compartida.
- A continuación se define la variable `rutaAlmacenamiento`, como su nombre lo indica contiene la ruta donde será depositada la entidad compartida transmitida.
- Posteriormente se copia el contenido de `params [0]` a la variable rutaAlmacenamiento. El parámetro `params [0]` contiene la ruta del contenedor destino de las entidades compartidas.
- Posteriormente se concatena el contenido de `params [1]` a la variable rutaAlmacenamiento. El parámetro `params [1]` contiene el nombre de la entidad compartida transmitida.
- A continuación se abre un flujo de escritura en la ruta contenida en la variable `rutaAlmacenamiento` y se genera el archivo con el nombre también almacenado en dicha variable. Dicho archivo es la entidad transmitida al sitio de trabajo.

6.3.3. Estado compartido de los recursos (parámetros para el control de la distribución)

Los mecanismos presentados anteriormente sirven para la comunicación entre un sitio de almacenamiento y un sitio participante. Para que dichos sitios cambién de rol (cliente-servidor a servidor-cliente) se agregan las banderas y comparaciones necesarias. De igual forma dichos mecanismos se pueden aplicar a una comunicación entre sitios de almacenamiento (red extendida), para tal situación se implementan otras comparaciones y banderas sobre el estado actual del sistema.

Para establecer en que situación un sitio realiza la función de sitio de trabajo o sitio de almacenamiento se le agrega al `mecanismo de transmisión de entidades compartidas y adaptadas` (ver figura 6.23) un módulo que proporciona las `banderas de estado` para así determinar el estado actual del sitio de trabajo. Dichas banderas son: 1) `CRL [4] = "CRL"` la cual representa al cliente de la red local 2) `SRL [4] = "SRL"` representa al servidor de la red local 3) `CRE [4] = "CRE"` representa al cliente de la red extendida y 4) `SRE [4] = "SRE"` representa al servidor de la red extendida.

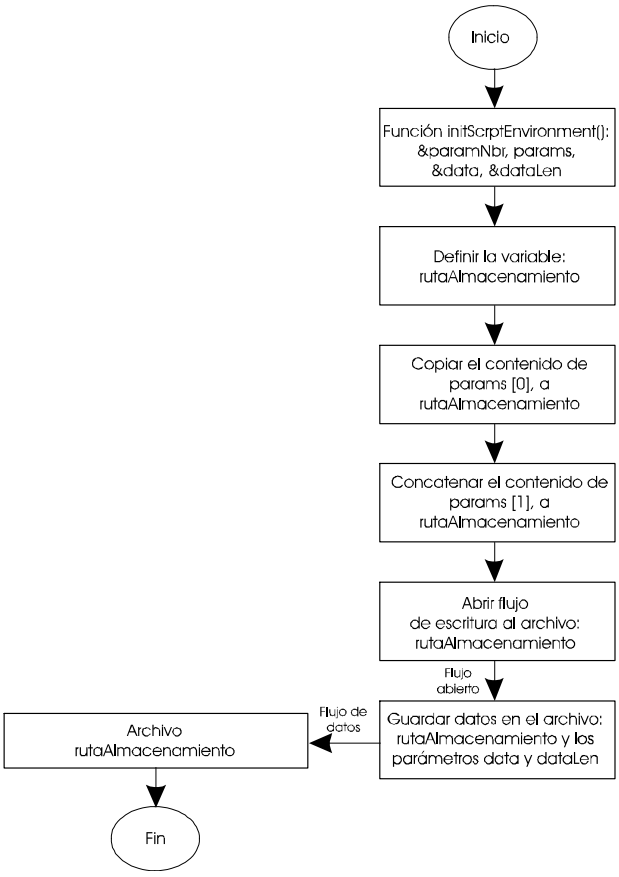

Figura 6.24: Recepción de las entidades compartidas adaptadas

La bandera en turno será almacenada en el arreglo `params [2]`. El parámetro `params [2]` de la función `initScriptEnvironment` es utilizado para transmitir y evaluar uno de los dos estados del sitio de trabajo o del sitio de almacenamiento. Las comparaciones utilizadas son: `if(strcmp(params [2], CRL) == 0)` para que la funcionalidad del sitio sea la de sitio de trabajo, la comparación para que funcione como sitio de almacenamiento sera: `if(strcmp(params [2], SRL) == 0)` y para sitio participante o de almacenamiento en la red extendida son de la siguiente forma respectivamente: `if(strcmp(params [2], CRE) == 0)` y `if(strcmp(params [2], SRE) == 0)`.

Una vez explicadas las banderas de estado, se debe mencionar que el mecanismo de transmisión debe ser habilitado por la bandera `SRL y SRE` dependiendo de la situación (red local o red extendida), así de igual forma, el mecanismo de recepción debe ser habilitado por las banderas `CRL Y CRE` respectivamente.

En la tabla siguiente se muestran las combinaciones iniciales de las banderas de estado en base al tipo del sitio.

sitio de almacenamiento	**sitio participante**
SRL	CRL
sitio de almacenamiento 1	**sitio de almacenamiento 2**
SRE	CRE
CRE	SRE

En la situación de que el sitio sea un sitio de almacenamiento, entonces la bandera de estado debe ser `SRL` y en el sitio de trabajo debe ser `CRL`. En caso de que la transmisión de entidades sea entre dos sitios de almacenamiento: el sitio de almacenamiento 1 debe tener la bandera de estado en `SRE` y el sitio de almacenamiento 2 en `CRE` y si la transmisión se realizara en sentido contrario, las etiquetas deben estar en `SRE` para el 2 y `CRE` para el 1.

6.4. Pruebas y resultados

Las pruebas realizadas tienen el objetivo de verificar el funcionamiento del sistema en la adaptación de las entidades compartidas en base a los siguientes aspectos: las capacidades del sitio participante, los derechos de acceso del usuario sobre las entidades, la decisión del usuario de recibir o no determinada entidad y el tamaño de las entidades compartidas.

Otros aspectos que fueron tomados en cuenta para llevar acabo la adaptación: los límites de los conjuntos difusos preestablecidos para cada variable lingüística y las categorías multimedia.

Finalmente los resultados obtenidos estan dados por: las reglas de inferencia y la versión de las entidades compartidas resultantes.

En esta sección son mostradas varias tablas que presentan los aspectos anteriormente mencionados. Primero se muestra como son evaluadas las entidades compartidas en un sitio de almacenamiento, a continuación se realiza dicha evaliuación con

tres sitios de trabajo. Las ecuaciones utilizadas es este apartado son explicadas en la sección 6.2.2.

6.4.1. Sitio de almacenamiento

En el actual escenario de dispone de un sitio de almacenamiento con las siguientes características:

- **velocidad del procesador**: 2400 MHz,
- **espacio disponible en el disco duro**: 102400 MB,
- **tamaño de la RAM**: 2048 MB,
- **resolución horizontal de pantalla**: 1280 píxeles.

Los límites de los conjuntos difusos definidos para cada una de las variables lingüísticas son:

- **velocidad del procesador**: 500, 1000, 1500 y 2000 MHz,
- **espacio disponible en el disco duro**: 2000, 4000, 6000 y 8000 MB,
- **tamaño de la RAM**: 250, 500, 750 y 1000 MB,
- **resolución horizontal de pantalla**: 500, 1000, 1500 y 2000 píxeles,
- **tamaño de la entidad compartida**: 1250, 2500, 3750 y 5000 Kbyte.

Se procede a evaluar la entrada precisa (valor de una característica del dispositivo) de cada variable lingüística (nombre de la característica del dispositivo). Dicha evaluación determina a cual de los siete subconjuntos difusos pertenece cada entrada precisa. En el actual escenario, las entradas precisas: la velocidad del procesador, el espacio disponible en el disco duro y el tamaño de la RAM se encuentran dentro del septimo subconjunto (SepSD). La entrada precisa de la resolución horizontal de pantalla se ubica en el cuarto y quinto subconjunto difuso (CSD y QSD). A continuación se muestra como se obtienen los grados de membresía aplicando las ecuaciones definidas en la sección 6.2.2.

Siendo $A = 1$ y EP (entrada precisa) corresponde al valor real de una variable lingüística.

- La velocidad del procesador, el espacio disponible en el disco duro y el tamaño de la RAM corresponden al séptimo subconjunto difuso (SepSD):

  SepSD:

$$\mu(EP) = A \quad EP > LSSD$$
$$\mu(2400) = 1 \quad 2400 > 2000$$

- La resolución horizontal de pantalla corresponde al cuarto y quinto subconjunto difuso (CSD y QSD).

  CSD:

$$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(1280) = 1(1500 - 1280)/(1500 - 1000) = 0.44 \quad 1000 < 1280 \leq 1500$$

- QSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(1280) = 1(1000 - 1280)/(1000 - 1500) = 0.56 \quad 1000 < 1280 \leq 1500$$

En la tabla se encuentran tabulados los resultados del procedimiento de fuzzificación sobre las entradas precisas de las capacidades del sitio participante.

Capacidades del sitio de almacenamiento		**conjuntos difusos**			
variable lingüística	**entrada precisa**	**B**	**M**	**A**	**MA**
velocidad del procesador (MHz)	2400	-	-	-	1
espacio disponible en el disco duro (MB)	102400	-	-	-	1
tamaño de la RAM (MB)	2048	-	-	-	1
resolución horizontal de pantalla	1280	-	0.44	0.56	-

Las pruebas realizadas están basadas en las cinco categorías multimedia definidas en el sistema de distribución de entidades y en cada una de estas se establecen cinco entidades de ejemplo. Los campos que integran la tabla son:

- **categoría**: contiene las cinco categorías multimedia definidas: texto, imagen, animación, audio y vídeo.
- **nombre de la entidad**: dependiendo de la categoría se utilizaron entidades con diferentes extensiones.
- **derechos de acceso por usuario**: son los derechos de acceso que tienen los usuarios sobre cada una de las entidades compartidas.
  - E = Escritura
  - L = Lectura
  - - = No aplica o Nulo

Sitio de almacenamiento				
categoría	**nombres de las entidades**	**derechos de acceso por usuario**		
texto	**entidades texto**	**usuario A**	**usuario B**	**usuario C**
	documento1.c	E	L	L
	documento2.c	L	E	-
	documento3.java	-	L	E
	documento4.conf	E	L	L
	documento5.txt	L	E	L
imagen	**entidades imagen**	**usuario A**	**usuario B**	**usuario C**
	imagen1.jpeg	L	L	E
	imagen2.bmp	E	-	L
	imagen3.jpeg	-	E	L
	imagen4.jpeg	-	L	E
	imagen5.jpeg	E	L	L
animación	**entidades animación**	**usuario A**	**usuario B**	**usuario C**
	animacion1.gif	E	L	L
	animacion2.gif	E	L	-
	animacion3.gif	E	L	-
	animacion4.gif	L	-	E
	animacion5.gif	-	E	L
audio	**entidades audio**	**usuario A**	**usuario B**	**usuario C**
	audio1.avi	-	E	L
	audio2.wav	-	L	E
	audio3.wav	L	E	L
	audio4.midi	E	L	L
	audio5.midi	E	-	L
vídeo	**entidades vídeo**	**usuario A**	**usuario B**	**usuario C**
	video1.jpgm	E	-	L
	video2.mpeg	L	L	E
	video3.asx	-	E	L
	video4.avi	E	L	L
	video5.movie	L	L	E

En la siguiente tabla se expone la decisión del usuario sobre cada una de las entidades compartidas, además se muestran los valores difusos del tamaño de cada entidad compartida. Los campos que integran esta tabla son:

- **Decisión del usuario**: el usuario decide que entidades quiere recibir, siempre y cuando tenga los derechos de acceso necesarios.
  - A = Aceptado
  - R = Rechazado

- **Tamaño**: se muestra el tamaño en Kbytes de las entidades, además de los límites definidos en los conjuntos difusos utilizados para la evaluación del tamaño de cada una de las entidades compartidas. este campo contiene la entrada precisa y como se puede observar, la mayoría de las entidades están ubicadas en el conjunto `bajo (B)`.

Se procede a evaluar la entrada precisa (valor del tamaño de cada entidad compartida) de cada variable lingüística (tamaño de la entidad compartida). Dicha evaluación determina a cual de los siete subconjuntos difusos pertenece cada entrada precisa. En el actual escenario, la mayoría de las entradas precisas sobre el tamaño de las entidades compartidas se encuentran en el primer subconjunto (PSD), dos más en el septimo (SepSD) y por último otros dos en el cuarto y quinto (CSD y QSD). A continuación se muestra como se obtienen los grados de membresía aplicando las ecuaciones definidas en la sección 6.2.2.

Siendo $A = 1$ y EP (entrada precisa) corresponde al valor real de una variable lingüística.

La velocidad del procesador, el espacio disponible en el disco duro y el tamaño de la RAM corresponden al séptimo subconjunto difuso (SepSD):

- Los siguientes archivos corresponden al primer subconjunto difuso (PSD): documento1.c, documento2.c, documento3.java, documento4.conf documento5.txt, imagen1.jpeg, imagen3.jpeg, imagen4.jpeg, imagen5.jpeg, animación1.gif, animació2.gif, animación3.gif, animación4.gif, animación5.gif, audio1.avi, audio2. wav, audio3.wav, audio4.midi, audio5.midi, video2.mpeg y video4.avi.

  PSD:

$$\mu(EP) = A \quad 0 < EP \leq LPSD$$
$$\mu(160) = 1 \quad 0 < 160 \leq 1250$$

- Los archivos image2.bmp y video5.movie corresponden al septimo conjunto difuso (SepSD):

  SepSD:

$$\mu(EP) = A \quad EP > LSSD$$
$$\mu(6749) = 1 \quad 6749 > 5000$$

- El archivo video1 corresponde al cuarto y quinto subconjunto (CSD y QSD):

  CSD:

$$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(3544) = 1(3750 - 3544)/(3750 - 2500) = 0.1648 \quad 2500 < 3544 \leq 3750$$

- QSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(3544) = 1(2500 - 3544)/(2500 - 3750) = 0.8352 \quad 2500 < 3544 \leq 3750$$

- El archivo video3 corresponde al cuarto y quinto subconjunto (CSD y QSD):

  CSD:

$$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(2560) = 1(3750 - 2560)/(3750 - 2500) = 0.952 \quad 2500 < 2560 \leq 3750$$

- QSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(2560) = 1(2500 - 2560)/(2500 - 3750) = 0.048 \quad 2500 < 2560 \leq 3750$$

En la siguiente tabla se encuentran tabulados la decisión del usuario y los resultados del procedimiento de fuzzificación. Esta tabla es la consecución de la anterior

Sitio de almacenamiento

decisión del usuario			**tamaño de las entidades compartidas conjuntos difusos (1250-2500-3750-5000)**				
usuario A	**usuario B**	**usuario C**	**tamaño**	**B**	**M**	**A**	**MA**
A	R	A	160	1	-	-	-
R	A	-	6	1	-	-	-
-	A	A	3	1	-	-	-
A	A	A	14	1	-	-	-
A	A	R	1	1	-	-	-
usuario A	**usuario B**	**usuario C**	**tamaño**	**B**	**M**	**A**	**MA**
A	R	R	174	1	-	-	-
A	-	R	6749	-	-	-	1
-	A	R	116	1	-	-	-
-	A	A	115	1	-	-	-
A	R	A	42	1	-	-	-
usuario A	**usuario B**	**usuario C**	**tamaño**	**B**	**M**	**A**	**MA**
A	R	R	52	1	-	-	-
A	A	-	11	1	-	-	-
A	A	-	13	1	-	-	-
R	-	A	21	1	-	-	-
-	A	A	225	1	-	-	-
usuario A	**usuario B**	**usuario C**	**tamaño**	**B**	**M**	**A**	**MA**
-	A	R	22	1	-	-	-
-	R	A	414	1	-	-	-
A	A	R	116	1	-	-	-
A	R	A	24	1	-	-	-
A	-	R	40	1	-	-	-

usuario A	**usuario B**	**usuario C**	**tamaño**	**B**	**M**	**A**	**MA**
A	-	A	3544	-	0.1648	0.8352	-
A	A	A	1024	1	-	-	-
-	A	A	2560	-	0.952	0.048	-
R	A	R	848	1	-	-	-
A	A	A	5233	-	-	-	1

En la tabla siguiente se muestran las reglas de inferencia y la versión de las entidades compartidas que han sido elegidas por el sistema para ser enviadas a los sitios participantes.

Sitio de almacenamiento

reglas de inferencia y versión de las entidades compartidas

servidor	usuario A	usuario B	usuario C
regla 1213 versión 3	regla 1150 versión 1	-	regla 1120 versión 3
regla 1213 versión 3	-	regla 1201 versión 3	-
regla 1213 versión 3	-	regla 1201 versión 3	regla 1120 versión 3
regla 1213 versión 3	regla 1150 versión 1	regla 1201 versión 3	regla 1120 versión 3
regla 1213 versión 3	regla 1150 versión 1	regla 1201 versión 3	-

servidor	usuario A	usuario B	usuario C
regla 1213 versión 3	regla 1150 versión 1	-	-
regla 1213 versión 3	regla 1152 substituto	-	-
regla 1213 versión 3	-	regla 1201 versión 3	-
regla 1213 versión 3	-	regla 1201 versión 3	regla 1120 versión 3
regla 1213 versión 3	regla 1150 versión 1	-	regla 1120 versión 3

servidor	usuario A	usuario B	usuario C
regla 1213 versión 3	regla 1150 versión 1	-	-
regla 1213 versión 3	regla 1150 versión 1	regla 1201 versión 3	-
regla 1213 versión 3	regla 1150 versión 1	regla 1201 versión 3	-
regla 1213 versión 3	-	-	regla 1120 versión 3
regla 1213 versión 3	-	regla 1201 versión 3	regla 1120 versión 3

servidor	usuario A	usuario B	usuario C
regla 1213 versión 3	-	regla 1201 versión 3	-
regla 1213 versión 3	-	-	regla 1120 versión 3
regla 1213 versión 3	regla 1150 versión 1	regla 1201 versión 3	-
regla 1213 versión 3	regla 1150 versión 1	-	regla 1120 versión 3
regla 1213 versión 3	regla 1150 versión 1	-	-

servidor	usuario A	usuario B	usuario C
regla 1215 versión 3	regla 1152 substituto	-	regla 1122 substituto
regla 1213 versión 3	regla 1150 versión 1	regla 1201 versión 3	regla 1120 versión 3
regla 1214 versión 3	-	regla 1202 versión 2	regla 1121 versión 1
regla 1213 versión 3	-	regla 1201 versión 3	-
regla 1215 versión 3	regla 1152 substituto	regla 1203 versión 1	regla 1122 substituto

A continuación se muestran tres ejemplos de sitios participantes y la distribución de entidades compartidas y adaptadas tomando como base los parámetros definidos al principio de esta sección y que fueron explicados en el ejemplo del sitio de almacenamiento.

6.4.2. Primer sitio participante

Se dispone de un sitio de de trabajo con las siguientes características:

- **velocidad del procesador**: 1800 MHz,
- **espacio disponible en el disco duro**: 2300 MB,
- **tamaño de la RAM**: 512 MB,
- **resolución horizontal de pantalla**: 1280 píxeles.

Los límites de los conjuntos difusos definidos para cada una de las variables lingüísticas son:

- **velocidad del procesador**: 500, 1000, 1500 y 2000 MHz,
- **espacio disponible en el disco duro**: 2000, 4000, 6000 y 8000 MB,
- **tamaño de la RAM**: 250, 500, 750 y 1000 MB,
- **resolución horizontal de pantalla**: 500, 1000, 1500 y 2000 píxeles,

Se procede a evaluar la entrada precisa (valor de una característica del dispositivo) de cada variable lingüística (nombre de la característica del dispositivo). Dicha evaluación determina a cual de los siete subconjuntos difusos pertenece cada entrada precisa. En el actual escenario la entrada precisa de la velocidad del procesador se encuentran dentro del sexto subconjunto (SSD). La entrada precisa del espacio disponible en el disco duro se ubica en el segundo y tercer subconjunto (SSD y TSD). La entrada precisa del tamaño de la RAM y la resolución horizontal de pantalla se encuentran en el cuarto y quinto subconjunto (CSD y QSD). A continuación se muestra como se obtienen los grados de membresía aplicando las ecuaciones definidas en la sección 6.2.2.

Siendo $A = 1$ y EP (entrada precisa) corresponde al valor real de una variable lingüística.

- La velocidad del procesador corresponde al sexto subconjunto difuso (SSD):

  SSD:

$$\mu(EP) = A \quad LCQSD < EP \leq LSSD$$
$$\mu(1800) = 1 \qquad 1500 < 1800 \leq 2000$$

- El espacio disponible en el disco duro corresponde al segundo y tercer subconjunto difuso (SSD y TSD):

  SSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LPSD) \quad LPSD < EP \leq LSTSD$$
$$\mu(2300) = 1(4000 - 2300)/(4000 - 2000) = 0.85 \qquad 2000 < 2300 \leq 4000$$

- TSD:

$$\mu(EP) = A(LPSD - EP)/(LPSD - LSTSD) \quad LPSD < EP \leq LSTSD$$
$$\mu(2300) = 1(2000 - 2300)/(2000 - 4000) = 0.15 \quad 2000 < 2300 \leq 4000$$

- El tamaño de la RAM corresponde al cuarto y quinto subconjunto difuso (CSD y QSD):

  CSD:

$$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(512) = 1(750 - 512)/(750 - 500) = 0.952 \quad 500 < 512 \leq 750$$

- QSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(512) = 1(500 - 512)/(500 - 750) = 0.048 \quad 500 < 512 \leq 750$$

- La resolución horizontal de pantalla corresponde al cuarto y quinto subconjunto difuso (CSD y QSD):

  CSD:

$$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(1280) = 1(1500 - 1280)/(1500 - 1000) = 0.44 \quad 1000 < 1280 \leq 1500$$

- QSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(1280) = 1(1000 - 1280)/(1000 - 1500) = 0.56 \quad 1000 < 1280 \leq 1500$$

En la tabla se muestran los resultados del procedimiento de fuzzificación sobre las entradas precisas de las capacidades del sitio participante.

capacidades del primer sitio participante		**conjuntos difusos**			
variable lingüística	**entrada precisa**	**B**	**M**	**A**	**MA**
velocidad del procesador (MHz)	1800	-	-	1	-
espacio disponible en el disco duro (MB)	2300	0.85	0.15	-	-
tamaño de la RAM (MB)	512	-	0.952	0.048	-
resolución horizontal de pantalla	1280	-	0.44	0.56	-

Primer sitio participante		
categoría	**nombres de las entidades**	**derechos de acceso por usuario**
texto	**entidades texto** documento1.txt documento2.html documento3.conf documento4.c documento5.asm	**usuario A** E L N E L
imagen	**entidades imagen** imagen1.bmp imagen2.jpg imagen3.png imagen4.tiff imagen5.ico	**usuario A** L E N N E
animación	**entidades animación** animacion1.gif animacion2.fla animacion3.swf animacion4. animacion5.	**usuario A** E E E L N
audio	**entidades audio** audio1.au audio2.snd audio3.mp3 audio4.wav audio5.midi	**usuario A** N N L E E
vídeo	**entidades vídeo** video1.jpgm video2.mpeg video3.asx video4.avi video5.movie	**usuario A** E L N E L

Primer sitio participante	
decisión del usuario	**versión de las entidades compartidas**

usuario A	**versiones texto**
A	versión 1
R	-
-	-
A	versión 1
A	versión 1

usuario A	**versiones imagen**
A	versión 1
A	substituto
-	-
-	-
A	versión 1

usuario A	**versiones animación**
A	versión 1
A	versión 1
A	versión 1
R	-
-	-

usuario A	**versiones audio**
-	-
-	-
A	versión 1
A	versión 1
A	versión 1

usuario A	**versiones vídeo**
A	substituto
A	versión 1
-	-
R	-
A	substituto

6.4.3. Segundo sitio participante

Se dispone de un sitio de de trabajo con las siguientes características:

- **velocidad del procesador**: 1639 MHz,
- **espacio disponible en el disco duro**: 39936 MB,
- **tamaño de la RAM**: 480 MB,
- **resolución horizontal de pantalla**: 800 píxeles.

Los límites de los conjuntos difusos definidos para cada una de las variables lingüísticas son:

- **velocidad del procesador**: 500, 1000, 1500 y 2000 MHz,
- **espacio disponible en el disco duro**: 2000, 4000, 6000 y 8000 MB,
- **tamaño de la RAM**: 250, 500, 750 y 1000 MB,
- **resolución horizontal de pantalla**: 500, 1000, 1500 y 2000 píxeles,

Se procede a evaluar la entrada precisa (valor de una característica del dispositivo) de cada variable lingüística (nombre de la característica del dispositivo). Dicha evaluación determina a cual de los siete subconjuntos difusos pertenece cada entrada precisa. En el actual escenario la entrada precisa de la velocidad del procesador se encuentra en el sexto subconjunto (SSD). La entrada precisa del espacio disponible en el disco duro se ubica en el septimo subconjunto (SepSD). La entrada precisa del tamaño de la RAM y la resolución horizontal de pantalla se encuentran en el segundo y tercer subconjunto (SSD y TSD). A continuación se muestra como se obtienen los grados de membresía aplicando las ecuaciones definidas en la sección 6.2.2.

Siendo $A = 1$ y EP (entrada precisa) corresponde al valor real de una variable lingüística.

- La velocidad del procesador corresponde al sexto subconjunto difuso (SSD):

  SSD:

$$\begin{array}{ll} \mu(EP) = A & LCQSD < EP \leq LSSD \\ \mu(1639) = 1 & 1500 < 1639 \leq 2000 \end{array}$$

- El espacio disponible en el disco duro corresponde al septimo subconjunto difuso (SepSD):

  SepSD:

$$\begin{array}{ll} \mu(EP) = A & EP > LSSD \\ \mu(39936) = 1 & 39936 > 8000 \end{array}$$

- El tamaño de la RAM corresponde al segundo y tercer subconjunto difuso (SSD y TSD):

  SSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LPSD) \quad LPSD < EP \leq LSTSD$$
$$\mu(480) = 1(500 - 480)/(500 - 250) = 0.08 \quad 250 < 480 \leq 500$$

- TSD:

$$\mu(EP) = A(LPSD - EP)/(LPSD - LSTSD) \quad LPSD < EP \leq LSTSD$$
$$\mu(480) = 1(250 - 480)/(250 - 500) = 0.92 \quad 250 < 480 \leq 500$$

- La resolución horizontal de pantalla corresponde al segundo y tercer subconjunto difuso (SSD y TSD):

  SSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LPSD) \quad LPSD < EP \leq LSTSD$$
$$\mu(800) = 1(1000 - 800)/(1000 - 500) = 0.4 \quad 500 < 800 \leq 1000$$

- TSD:

$$\mu(EP) = A(LPSD - EP)/(LPSD - LSTSD) \quad LPSD < EP \leq LSTSD$$
$$\mu(800) = 1(500 - 800)/(500 - 1000) = 0.6 \quad 500 < 800 \leq 1000$$

A continuación se presenta la tabla con los resultados del procedimiento de fuzzificación sobre las entradas precisas de las capacidades del sitio participante.

capacidades del segundo sitio participante		**conjuntos difusos**			
variable lingüística	**entrada precisa**	**B**	**M**	**A**	**MA**
velocidad del procesador (MHz)	1639	-	-	1	-
espacio disponible en el disco duro (MB)	39936	-	-	-	1
tamaño de la RAM (MB)	480	0.08	0.92	-	-
resolución horizontal de pantalla	800	0.4	0.6	-	-

Segundo sitio participante		
categoría	**nombre del recurso**	**derechos de acceso por usuario**
texto	**entidades texto**	**usuario B**
	documento1.txt	L
	documento2.html	E
	documento3.conf	L
	documento4.c	L
	documento5.asm	E
imagen	**entidades imagen**	**usuario B**
	imagen1.bmp	L
	imagen2.jpg	N
	imagen3.png	E
	imagen4.tiff	L
	imagen5.ico	L
animación	**entidades animación**	**usuario B**
	animacion1.gif	L
	animacion2.fla	L
	animacion3.swf	L
	animacion4.	N
	animacion5.	E
audio	**entidades audio**	**usuario B**
	audio1.au	E
	audio2.snd	L
	audio3.mp3	E
	audio4.wav	L
	audio5.midi	N
vídeo	**entidades vídeo**	**usuario B**
	video1.jpgm	N
	video2.mpeg	L
	video3.asx	E
	video4.avi	L
	video5.movie	L

Segundo sitio participante	
decisión del usuario	**versión de las entidades compartidas**
usuario B	**versiones texto**
R	-
A	versión 3
A	versión 3
A	versión 3
A	versión 3
usuario B	**versiones imagen**
R	-
-	-
A	versión 3
A	versión 3
R	-
usuario B	**versiones animación**
R	-
A	versión 3
A	versión 3
-	-
A	versión 3
usuario B	**versiones audio**
A	versión 3
R	-
A	versión 3
R	-
-	-
usuario B	**versiones vídeo**
-	-
A	versión 3
A	versión 2
A	versión 3
A	versión 1

6.4.4. Tercer sitio participante

Se dispone de un sitio de de trabajo con las siguientes características:

- **velocidad del procesador**: 1229 MHz,
- **espacio disponible en el disco duro**: 13210 MB,
- **tamaño de la RAM**: 512 MB,
- **resolución horizontal de pantalla**: 1152 píxeles.

Los límites de los conjuntos difusos definidos para cada una de las variables lingüísticas son:

- **velocidad del procesador**: 500, 1000, 1500 y 2000 MHz,
- **espacio disponible en el disco duro**: 2000, 4000, 6000 y 8000 MB,
- **tamaño de la RAM**: 250, 500, 750 y 1000 MB,
- **resolución horizontal de pantalla**: 500, 1000, 1500 y 2000 píxeles,

Se procede a evaluar la entrada precisa (valor de una característica del dispositivo) de cada variable lingüística (nombre de la característica del dispositivo). Dicha evaluación determina a cual de los siete subconjuntos difusos pertenece cada entrada precisa. En el actual escenario la entrada precisa de la velocidad del procesador se encuentra en el cuarto y quinto subconjunto (CSD y QSD). La entrada precisa del espacio disponible en el disco duro se ubica en el septimo subconjunto (SepSD). La entrada precisa del tamaño de la RAM y la resolución horizontal de pantalla se encuentra en el cuarto y quinto subconjunto (CSD y QSD). A continuación se muestra como se obtienen los grados de membresía aplicando las ecuaciones definidas en la sección 6.2.2.

Siendo $A = 1$ y EP (entrada precisa) corresponde al valor real de una variable lingüística.

- La velocidad del procesador corresponde al cuarto y quinto subconjunto difuso (CSD y QSD):

  CSD:

$$\begin{array}{ll} \mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) & LSTSD < EP \leq LCQSD \\ \mu(1229) = 1(1500 - 1229)/(1500 - 1000) = 0.542 & 1000 < 1229 \leq 1500 \end{array}$$

- QSD:

$$\begin{array}{ll} \mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) & LSTSD < EP \leq LCQSD \\ \mu(1229) = 1(1000 - 1229)/(1000 - 1500) = 0.458 & 1000 < 1229 \leq 1500 \end{array}$$

- El espacio disponible en el disco duro corresponde al septimo subconjunto difuso (SepSD):

  SepSD:

$$\mu(EP) = A \quad EP > LSSD$$
$$\mu(1229) = 1 \quad 13210 > 8000$$

- El tamaño de la RAM corresponde al cuarto y quinto subconjunto difuso (CSD y QSD):

  CSD:

$$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(512) = 1(750 - 512)/(750 - 500) = 0.952 \quad 500 < 512 \leq 750$$

- QSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(512) = 1(500 - 512)/(500 - 750) = 0.048 \quad 500 < 512 \leq 750$$

- La resolución horizontal de pantalla corresponde al cuarto y quinto subconjunto difuso (CSD y QSD):

  CSD:

$$\mu(EP) = A(LCQSD - EP)/(LCQSD - LSTSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(1152) = 1(1500 - 1152)/(1500 - 1000) = 0.696 \quad 1000 < 1152 \leq 1500$$

- QSD:

$$\mu(EP) = A(LSTSD - EP)/(LSTSD - LCQSD) \quad LSTSD < EP \leq LCQSD$$
$$\mu(1152) = 1(1000 - 1152)/(1000 - 1500) = 0.304 \quad 1000 < 1152 \leq 1500$$

La tabla siguiente contiene los resultados del procedimiento de fuzzificación sobre las entradas precisas de las capacidades del sitio participante.

capacidades del tercer sitio participante		**conjuntos difusos**			
variable lingüística	**entrada precisa**	**B**	**M**	**A**	**MA**
velocidad del procesador (MHz)	1229	-	0.542	0.458	-
espacio disponible en el disco duro (MB)	13210	-	-	-	1
tamaño de la RAM (MB)	512	-	0.952	0.048	-
resolución horizontal de pantalla	1152	-	0.696	0.304	-

Tercer sitio participante		
categoría	**nombre del recurso**	**derechos de acceso por usuario**
texto	**entidades texto** documento1.txt documento2.html documento3.conf documento4.c documento5.asm	**usuario C** L N E L L
imagen	**entidades imagen** imagen1.bmp imagen2.jpg imagen3.png imagen4.tiff imagen5.ico	**usuario C** E L L E L
animación	**entidades animación** animacion1.gif animacion2.fla animacion3.swf animacion4. animacion5.	**usuario C** L N N E L
audio	**entidades audio** audio1.au audio2.snd audio3.mp3 audio4.wav audio5.midi	**usuario C** L E L L L
vídeo	**entidades vídeo** video1.jpgm video2.mpeg video3.asx video4.avi video5.movie	**usuario C** L E L L E

Tercer sitio participante	
decisión del usuario	**versión de las entidades compartidas**
usuario C	**versiones texto**
A	versión 3
-	-
A	versión 3
A	versión 3
R	-
usuario C	**versiones imagen**
R	-
R	-
R	-
A	versión 3
A	versión 3
usuario C	**versiones animación**
R	-
-	-
-	-
A	versión 3
A	versión 3
usuario C	**versiones audio**
R	-
A	versión 3
R	-
A	versión 3
R	-
usuario C	**versiones vídeo**
A	substituto
A	versión 3
A	versión 1
R	-
A	substituto

Capítulo 7

Conclusiones y trabajo futuro

Los movimientos sociales y científicos han generado y todavía generan un constante avance tecnológico y un mundo globalizado lo cual ha afectado y modificado el estilo de vida de los seres humanos. Conforme avanza el tiempo se van desarrollando herramientas informáticas que afectan el estilo de trabajo entre colaboradores geográficamente distribuidos. Dicha tendencia involucra el uso de grandes cantidades de información física (documentos, cartas) o digital (archivos, bases de datos).

Esta constante evolución de los ambientes de trabajo y de la tecnología asociada abrió nuevas posibilidades de interacción entre los usuarios que les permitan producir en grupos. Esta tecnología (aplicaciones o sistemas) que permite trabajar en grupo se le conoce como *groupware*. Así, siguiendo esta misma evolución, se desarrollo un nuevo dominio de investigación que es el Trabajo Colaborativo Asistido por Computadora (`TCAC` ó `CSCW` en inglés) que tiene por objetivo entender la manera en que la gente trabaja en grupos utilizando las tecnologías de comunicación e información y así proponer soluciones adecuadas.

En este trabajo, propusimos estudiar los problemas de infraestructura que dan soporte al trabajo cooperativo distribuido en la Web.

A continuación se describe el problema que se atacó, así como las contribuciones realizadas y algunas ideas sobre extensiones que se pueden diseñar e integrar en la solución propuesta: un sistema de distribución adaptable de entidades compartidas.

7.1. Resumen de la problemática

La problemática en este trabajo de tesis se enfoca en diseñar e implementar los mecanismos de una arquitectura de distribución de entidades compartidas y adaptadas para los sitios participantes involucrados en una sesión de colaboración entre personas geográficamente distribuidas.

La funcionalidad de una arquitectura de distribución define la forma y las características de la infraestructura en que las entidades compartidas y los componentes de tratamiento de un sistema colaborativo están distribuidas o al contrario centralizadas en los sitios participantes de un proceso de colaboración. Las arquitecturas básicas

usualmente utilizadas son la centralizada y la totalmente replicada, aunque también se definieron otras como: la asímétrica, la híbrida y de múltiples servidores. Estas tres últimas se presentan como variaciones de las dos primeras, a las cuales se le agregan otras funciones que hacen la diferencia entre ellas. Se toma como precedente que las arquitecturas de distribución difieren en tres aspectos: 1) la eventual presencia y la representación de una entidad compartida en el sitio local, 2) el número de ejemplares de una entidad presentes en el sistema y 3) la posible movilidad de una entidad entre los sitios participantes.

Para diseñar una arquitectura de distribución, algunas limitaciones deben ser tomados en cuenta:

1. El dispositivo receptor no tenga las capacidades suficientes para recibir, almacenar y procesar una entidad. Por lo tanto se recibirá un representante (un descriptor que permite localizar y acceder a la entidad real) de dicha entidad, ya que se aplica el principio de todo o nada,

2. Otro inconveniente es que al guardar una réplica de cada entidad en un sitio participante se va incrementando el espacio utilizado en el disco duro de este sitio, aun cuando cada colaborador no requiera de alguna de las entidades o no tenga derecho de acceso.

Algunas de las desventajas que tienen los aspectos de la distribución en la arquitectura centralizada se refiere a que las entidades compartidas residen en un sitio único que actúa como un servidor. Esto implica que este sitio va a recibir y procesar solicitudes provenientes de otros sitios en los cuales la entidad es personificada por un representante (también conocido como substituto). La desventaja existente es que se genera un cuello de botella y la viabilidad del sistema está sensible a las potenciales fallas del servidor central o de los medios de comunicaciones. En la arquitectura replicada, la duplicación de una entidad remota en el sitio local incrementa: a) el rendimiento (*performance)* y la viabilidad del sistema colaborativo y b) la autonomía de cada colaborador. La desventaja es el aumento sensible a la eficiencia, ya que habrá un número mayor en la transmisión de mensajes.

Sin embargo, algunos de los sitios participantes de los colaboradores pueden tener bajas capacidades para almacenar, procesar y desplegar entidades compartidas, aunado a esto, si estas entidades son de gran tamaño para su almacenamiento y tratamiento en alguno de dichos sitios, entonces las arquitecturas de distribución tradicionales no proveen los mecanismos necesarios para determinar cuales entidades pueden ser o no procesadas por el sitio participante.

7.2. Resumen de las contribuciones

El sistema de distribución adaptable de entidades desarrollado tiene el objetivo de evaluar el sitio participante solicitante y determinar las entidades compartidas a enviar tomando como base las capacidades de procesamiento, almacenamiento y

despliegue. Este trabajo de investigación/prototipaje fue realizado en dos etapas: en la primera fase, se estudio de forma general los aspectos teóricos presentados en esta tesis y sobre los cuales se plasma el conocimiento adquirido y necesario para desarrollar el sistema de distribución adaptable. En la segunda fase, se estudiaron y se prototiparon los mecanismos dedicados al soporte del trabajo cooperativo en la Web, los cuales constituyen las aportaciones prácticas de este trabajo.

Del punto de los aspectos teóricos.

- **Análisis y caracterización de las arquitecturas de distribución**: se estudiaron las principales arquitecturas de distribución, además de las infraestructuras que utilizan escenas de distribución no flexibles y flexibles. Por último, se explicó la categorización de las entidades compartidas y los mecanismos de tratamiento considerando el lugar donde se almacenan, se utilizan y se ejecutan.

- **Aplicación de la lógica difusa y de los árboles de decisión para resolver el problema de evaluar las capacidades del sitio participante y el tamaño de las entidades compartidas**: se investigó la teoría de los conjuntos clásicos y difusos, así como algunos tipos de sistemas lógicos difusos y la teoría de árboles de decisión la cual se utilizó para generar las reglas de inferencia. Estas reglas constituyen la base de un motor de inferencia cuyo objetivo es determinar cuales son las entidades más adecuadas para un sitio participante.

- **Estudio y utilización de las arquitecturas y mecanismos de comunicación**: se analizarón de forma general las arquitecturas cliente-servidor y *peer to peer*, así como algunos servidores Web (Apache especialmente). Se realizó un estudio sobre el funcionamiento del protocolo HTTP, así también los modelos OSI y TCP/IP y por último algo sobre la teoría de los *scripts* CGI. Esta tecnología fue usada para diseñar e implementar el mecanismo de distribución de entidades.

Del punto de los desarrollos prácticos:

En el siguiente listado se especifican cada uno de los mecanismos diseñados e implementados para el sistema de distribución adaptable de entidades compartidas:

- **Mecanismo de adaptación y transmisión de entidades:** se generaron mecanismos para adaptar y transmitir entidades compartidas hacia los sitios participantes. La adaptación consistió en la evaluación de las capacidades del sitio participante y el tamaño de la entidad compartida para determinar su grado de membresía en los conjuntos difusos (bajo, medio y alto) y su posterior envío de los valores resultantes al motor de inferencia para que se determine la versión de la entidad más adecuada al sitio participante solicitante.

- **Mecanismo de selección de entidades:** se definió el mecanismo para que el usuario pueda escoger las entidades que quiere recibir durante una sesión colaborativa. Dicho mecanismo despliega en la pantalla una por una las entidades compartidas, de las cuales el usuario las acepta o rechaza.

- **Mecanismo de recuperación de las capacidades de los sitios cooperantes:** se desarrollo el mecanismo para obtener las capacidades del sitio participante: velocidad del procesador, espacio disponible en el disco duro, tamaño de la RAM y resolución horizontal de pantalla,

- **Mecanismo de selección de la versión adecuada de una entidad:** se diseño e implementó el mecanismo de selección de las versiones (ejemplares disminuidos de cada entidad) de entidades más adecuadas al sitio participante. Esta funcionalidad está implementada e incluye los siguientes mecanismos:

  - un mecanismo para determinar la existencia del software requerido al procesamiento de las entidades compartidas y al establecimiento de la categoría MIME a la que pertenece cada una de las entidades,
  - un mecanismo de fuzzificación de las capacidades del sitio participante y del tamaño de la entidad. La fuzzificación es la conversión de las capacidades y del tamaño de la entidad a valores difusos para ser utilizados en el motor de inferencia.
  - un motor de inferencia y un conjunto de reglas de inferencia para determinar que entidades serán enviadas al sitio participante.
  - la ampliación del mecanismo de transmisión de entidades de PIÑAS para enviar y recibir las entidades seleccionadas.

- **Mecanismo de distribución de entidades**: se desarrollo un mecanismo de replicación (envío y recepción) de las entidades compartidas entre los sitios cooperantes.

  Este mecanismo recibe como entradas la versiones de las entidades arrojadas por el motor de inferencia, para posteriormente enviarlas al sitio de trabajo.

7.3. Investigación futura

En seguida se presentan algunas mejoras y/o extensiones que se pueden realizar al sistema de distribución y adaptación de entidades compartidas que fue propuesto en este trabajo de investigación:

- El mecanismo de distribución desarrollado solo fue probado en dispositivos (PCs y Laptops) conectados a una red alámbrica. Una extensión muy interesante será adaptar este mecanismo para dar soporte a dispositivos móviles en redes inalámbricas (e.g., PDA). La diferencia entre el mecanismo actual y la extensión

propuesta sería la disminución de procesamiento, almacenamiento y desplieque en los dispositivos móviles utilizados por los colaboradores.

Otra limitación actual es que el sistema no determina las características de la entidad compartida a trasmitir considerando el ancho de banda de la red asociada al dispositivo sin importar que este sea móvil o fijo.

- La metodología utilizada para la definición de las reglas de inferencia es basada en árboles de decisión, así que una mejora es la reducción de dichas reglas que actualmente son de 1215. El total de reglas esta dado por el producto $3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 5$, donde cada tres representa respectivamente a los conjuntos difusos (B, M y A) de las cuatro capacidades del dispositivo y del tamaño de la entidad compartida. Por último, el valor cinco corresponde a las categorías multimedia definidas (texto, imagen, animación, audio y video).

  Además de la metodología anterior, se pueden utilizar funciones adecuadas para obtener mejores resultados como son el centro de gravedad y los centros promediados utilizados en los sistemas tipo Mamdani [Chahuara, 2005] (ver sección 3.1.4).

- Otra posibilidad para reducir el total de reglas de inferencia es desarrollar un modelo de clasificación parcial, en el cual se buscan características comunes entre dichas reglas sin tener que construir un modelo de predicción completo, como es presentado en la sección 3.2.1.

- Otro punto a tratar es implementar el soporte para múltiples archivos extensiones.lst (ver sección 6.2.5), ya que actualmente soporta un archivo único.

- La distribución de entidades se realiza entre dos sitios participantes que actúan alternativamente como cliente o como servidor: mientras el primero hace la función de cliente, el segundo actúa como servidor. La mejora sería diseñar e implementar un soporte dedicado que funcione con varios sitios participantes a la vez y que mantenga la funcionalidad antes mencionada.

- En el prototipo que se desarrolló, fueron propuestos mecanismos básicos para soportar la distribución/replicación de las entidades compartidas basados en la evaluación de las capacidades de los sitios cooperantes. Como complemento de eso, parece esencial implementar mecanismos para medir la disponibilidad de entidades compartidas y la administración de uso del espacio de almacenamiento en el que se evite la duplicación de entidades en el mismo sitio.

- En el sistema se definieron y utilizaron cuatro variables lingüísticas: la velocidad del procesador, el espacio disponible en el disco duro, el tamaño de la memoria RAM y la resolución horizontal de pantalla. La mejora sería ampliar el número de variables lingüísticas, e.g., el tiempo de acceso al disco duro, el número de núcleos del procesador y la resolución vertical de la pantalla, entre otros.

Se debe tomar en cuenta que el aumento de variables lingüísticas conlleva a un incremento de las reglas de inferencia si es que se siguen utilizando árboles de decisión. Si este fuera el caso, se debe buscar otra alternativa para definir las reglas de inferencia que determinen las versiones más adecuadas al sitio participante.

Capítulo 8

Apéndices

Anexos. En este capítulo se definen algunos conceptos básicos de la lógica difusa. Posteriormente, se muestran las tablas de verdad utilizadas en la generación de las reglas de inferencia que permiten determinar qué versión de una entidad compartida es la más adecuada para un sitio de trabajo. Por último, se explica cómo se compila, se instala, se configura y se ejecuta el servidor Apache.

8.1. Conceptos sobre la teoría de conjuntos difusos

Algunos conceptos importantes para comprender la teoría de la lógica difusa se describen a continuación:

- **Inteligencia artificial**: es una rama de la informática que se dedica al estudio de la simulación de la inteligencia utilizando programas para tratar de imitar la inteligencia de los seres vivos. La principal aplicación de esta rama es la creación de máquinas para la automatización de tareas que requieran un comportamiento inteligente.

- **Lógica clásica**: una proposición sólo admite dos valores: verdadero o falso, por lo que se dice que esta lógica es bivalente o binaria. Existen otras lógicas que soportan un tercer valor posible (lógica trivaluada) e incluso múltiples valores (lógica multivaluada). La lógica clásica sirve para explicar ciertos fenómenos y problemas, pero la mayoría de ellos en el ámbito teórico de la matemática.

- **Lógica multivaluada**: incluye sistemas lógicos que admiten varios valores de verdad posibles (e.g., bajo-medio-alto).

- **Lógica difusa o borrosa**: es una lógica multivaluada que se caracteriza por cuantificar la siguiente incertidumbre: si P es una proposición, se le puede asociar un número v(P) en el intervalo [0,1] tal que: 1) si v(P)=0, P es falso, 2) si v(P)=1, P es verdadero y 3) la veracidad de P aumenta con v(P), en la lógica clásica se

definirían dos tipos de personas, bajas y altas, siempre y cuando estén dentro de un rango de estatura, e.g., si una persona mide más de 1.75 m se dice que es alta, entonces todas las demás personas que tengan una estatura menor o igual a 1.75 m serán consideradas bajas. Sin embargo, mediante la lógica difusa se pueden definir tres o más conjuntos difusos: bajo, medio y alto, lo que significa que ahora toda persona que mida 1.75 m, tiene un grado de pertenencia a cada uno de estos conjuntos (e.g., Bajo=25 % Medio=50 % y Alto=25 %).

- **Sistemas difusos**: los sistemas difusos tienen su origen en la Inteligencia Artificial y a la vez forman parte de una tecnología de computación llamada *soft-computing* (computación flexible). La computación flexible engloba un conjunto de técnicas que tienen en común la robustez en el manejo de la información imprecisa e incierta, que existe en los problemas relacionados con el mundo real (e.g., toma de decisiones, clasificación y reconocimiento de formas). La computación flexible no garantiza la mejor solución, pero da la posibilidad de utilizar una serie de técnicas que son aplicables a muchos problemas, sin tener que modificar significativamente los sistemas que hagan uso de dichas técnicas difusas.

- **Conjuntos difusos**: la lógica difusa se deriva de la teoría de los conjuntos difusos, cuyo razonamiento se basa en la aproximación a la percepción humana: no todo es blanco o negro, existen varios tipos de grises, los cuales predominan en el pensamiento humano. Por lo tanto, se da mayor prioridad a la aproximación que a la precisión, la cual recibía toda la atención e importancia de la lógica clásica.

8.2. Servidor Apache

El procedimiento de compilación, instalación, configuración y ejecución del servidor Apache es explicado en las siguientes subsecciones:

8.2.1. Instalación y ejecución del servidor Apache

Existen dos formas de instalar el servidor Apache: 1) mediante la compilación del código fuente y 2) mediante un paquete binario adecuado al sistema operativo que se esté usando. En esta sección sólo se explica el primer método [Mateu, 2004].

Compilación a partir del código fuente

1. Primero se debe obtener la versión del servidor Apache en el portal http:// httpd.apache.org,

2. Se descomprime el archivo de instalación de Apache que se haya descargado, lo cual genera un directorio donde está localizado el código fuente del servidor,

3. Se accede a dicho directorio,

4. Se configura el código para compilarlo mediante el comando `./configure`, el cual dispone de varios parámetros para ajustar la compilación de Apache. En la tabla se muestran dichos parámetros:

Parámetro	Significado
–prefix	Directorio donde se quiere instalar Apache
–enable-modules	Módulos que hay que activar
=LISTA-MODULOS	
–enable-mods-shared	Módulos *shared* que hay que activar
=LISTA-MODULOS	
–enable-cache	Caché dinámico
–enable-disk-cache	Caché dinámico en disco
–enable-mem-cache	Módulo de cache de memoria
–enable-mime-magic	Determinación tipo MIME automática
–enable-usertrack	Seguimiento de sesión de usuario
–enable-proxy	Módulo Apache-proxy
–enable-proxy-connect	Módulo Apache-proxy para CONNECT
–enable-proxy-ftp	Módulo Apache-proxy para FTP
–enable-proxy-http	Módulo Apache-proxy HTTP
–enable-ssl	Soporte de SSL/TLS (mod ssl)
–enable-http	Manejo del protocolo HTTP
–enable-dav	Manejo del protocolo WebDAV
–disable-cgid	Soporte de CGI optimizado
–enable-cgi	Soporte de CGI
–disable-cgi	Soporte de CGI
–enable-cgid	Soporte de CGI optimizado
–enable-vhost-alias	Soporte de *host* virtuales

5. Se compila el código fuente configurado mediante el comando `make.` Para llevar acabo la compilación se requiere el compilador GNU gcc.

6. Por último, se instala en el directorio que se asignó durante la configuración (./configure). La instalación se lleva acabo mediante el uso del comando `make install`.

Ejecución del servidor Apache

1. Una vez que el servidor Apache ha sido instalado se procede a su ejecución,

2. Se accede al directorio donde Apache haya sido instalado,

3. Se accede al directorio *bin*,

4. Se ejecuta el comando `./apachectl start`, el cual permite iniciar el servidor.

Detención del servidor Apache

Para detener la ejecución del servidor se ejecuta el comando: `./apachectl stop`.

8.2.2. httpd.conf

La configuración de Apache se encuentra en el archivo `httpd.conf`, el cual se ubica en el directorio `conf` que, a su vez, está localizado en el directorio raíz del servidor. Apache incluye una configuración mínima que arranca el servidor en el puerto TCP 80, el cual es el puerto predeterminado.

Estructura del fichero de configuración

El archivo `httpd.conf` está dividido en tres bloques principales y fundamentales [Mateu, 2004]:

1. **Parámetros globales**: son parámetros generales que afectan a todo el servidor,
2. **Directivas de funcionamiento**: permiten el control o acceso a algún contenido mediante modificaciones en la configuraciones definidas en las secciones, las cuales están ubicadas en el archivo `httpd.conf`,
3. **Hosts virtuales**: Apache puede atender diversos sitios Web con un sólo servidor, mediante la creación de dominios virtuales en función de diversas direcciones IP o diversos nombres por IP.

El archivo `httpd.conf` también contiene secciones y directivas que permiten cambiar la configuración del servidor. Algunas de las secciones más importantes son [Mateu, 2004]:

- <**Directory**>: los parámetros que se encuentran dentro de esta sección sólo se aplicarán al directorio especificado y a sus subdirectorios;
- <**Files**>: los parámetros de configuración proporcionan control de acceso a los archivos por su nombre;
- <**Location**>: proporciona un control de acceso de los archivos por medio de una URL;
- <**IfModule**>: se aplican los parámetros si al arrancar el servidor, el módulo especificado se encuentra cargado con `LoadModule`;
- <**VirtualHost**>: los parámetros sólo se aplicarán a las peticiones que vayan dirigidas a este *host* (nombre del servidor o dirección IP y el puerto TCP).

Algunas de las **directivas** [Mateu, 2004] más importantes son:

- **ServerRoot**: especifica la ubicación del directorio raíz donde está instalado el servidor Web. A partir de aquí se ubican los programas que ejecutan y detienen el servidor, los archivos de configuración, los *scripts* CGI, los archivos HTML, etc.
- **Listen**: especifica en qué puerto se atenderán las peticiones. Por defecto se utiliza el puerto 80 de TCP, pero también permite especificar qué direcciones IP se utilizarán por si el servidor tuviese más de una.
- **LoadModule**: con esta directiva es posible cargar en el servidor los módulos adicionales de Apache.
- **ServerAdmin**: sirve para especificar la dirección de correo electrónico del administrador. Esta dirección puede aparecer como dirección de contacto en los mensajes de error para permitir a los usuarios notificar algún error al administrador.
- **ServerName**: aquí se especifica el nombre y el puerto TCP que el servidor utiliza para identificarse, aunque se puede determinar automáticamente. En caso de que el servidor no tenga un nombre DNS, se tiene que incluir su dirección IP.
- **DocumentRoot**: aquí se especifica el directorio donde se servirán los documentos. El directorio predeterminado es `htdocs`.
- **Alias**: las directivas `Alias` y `AliasMatch` permiten definir accesos a directorios que se encuentran fuera del `DocumentRoot`.

8.2.3. Configuración del servidor Apache

En esta sección se explica el procedimiento para instalar y configurar el servidor Web http-2.2.8.tar.gz. Los comandos para la descompresión, la compilación y la instalación son:

1. descompresión del servidor:

   `tar xvzf httpd-2.2.8.tar.gz`,

2. ruta donde se instalará Apache:

   `./configure --prefix=/home/directorio_usuario/Apache`,

3. compilación del servidor:

   `make`,

4. instalación del servidor:

   `make install`.

Para que el servidor funcione adecuadamente, se debe modificar las siguientes directivas en el archivo de configuración `httpd.conf` ubicado en el directorio `/home/directorio_usuario/Apache/conf/httpd.conf`. Dichas directivas se citan a continuación:

- **Listen**: especificación de la dirección IP o del nombre de dominio, además del puerto asignado al servidor.

Por ejemplo: Listen localhost:2008

- **User y Group**: especifican el nombre de usuario donde está instalado el servidor y el grupo al que pertenece dicho usuario.

Por ejemplo: *User* javier y *Group* estudiantes

- **ServerName**: se define el nombre del servidor y el puerto de acceso.

Por ejemplo: ServerName localhost:2008

- **ScriptsAlias y <Directory = “”>**: sirve para hacer un alias o referencia a un directorio que contenga *scripts*, con el fin de poder accederlos mediante una navegador.

ScriptAlias /cgi-bin/ “/home/directorio_usuario/Apache/cgi-bin/”
La parte `/cgi-bin/` es la que se modifica para hacer referencia a otro directorio.

Por ejemplo: ScriptAlias `/pinas/` “/home/directorio_usuario/Apache/cgi-bin/”
Posteriormente se activan los permisos de ejecución de los archivos cgi, los cuales están ubicados en el directorio /home/directorio_usuario/Apache/cgi-bin. Para cambiar los permisos de acceso, se utiliza el comando chmod +x *.

Por último, se inicia el servidor accediendo a la ruta `/home/directorio_usuario/Apache/bin/` y ejecutando el comando `./apachectl start`. Para detenerlo o reiniciarlo, se utilizan los comandos `./apachectl stop` y `./apachectl restart` respectivamente.

8.2.4. Compilación de la interfaz de comunicación

La interfaz de comunicación está integrada por los directorios `CGIappli` y `libwww`. En el directorio `CGIappli`, se ejecuta el *script* `mkf` para configurar las direcciones de los archivos de la interfaz de comunicación, en base al servidor y a la cuenta de usuario. Posteriormente, se utiliza el comando *make* el cual compilará toda la interfaz. A continuación, se copia el archivo *pscript* generado en la ruta `~Apache/cgi-bin/`. Por último, se ejecuta el *script* `essaipost`, el cual está ubicado en el directorio `/home/directorio_usuario/CGIappli/bin/`. Dicho *script* solicitará los siguientes datos: `nombre del host`, `número de puerto` y `nombre del archivo` a transferir. El

contenido del archivo será enviado al *host* donde debe estar instalado `pscript` y este a su vez lo regresa a `essaipost`, el cual lo almacena con el nombre de *back.out* en el directorio `CGIappli`.

En caso de modificar el número de parámetros de la interfaz, se debe ejecutar el *script make.interfaces*.

8.3. Tablas de verdad de las reglas difusas

En las tablas mostradas en este apartado, se especifican los valores de las reglas de inferencia que permiten determinar las entidades más adecuadas en base a las capacidades de los sitios de trabajo y del tamaño de la entidad evaluada. Las variables lingüísticas que son evaluadas y que forman parte de dichas reglas son: a) la velocidad del procesador, b) el espacio disponible en el disco duro, c) el tamaño de la memoria RAM, d) la resolución horizontal de la pantalla y e) el tamaño de la entidad compartida.

Las constantes: B, M y A contienen los valores con los que se compara a cada uno de los parámetros: X1, X2, X3, X4 y X5. Dichos parámetros poseen el valor representativo (1, 2 o 3) de los conjuntos difusos resultantes (Bajo, Medio o Alto) en el proceso de fuzzificación. Los primeros cuatro parámetros representan respectivamente a las cuatro variables lingüísticas de las capacidades y el quinto a la entidad compartida evaluada actualmente:

- B = Bajo = 1
- M = Mediano = 2
- A = Alto = 3

Los posibles resultados que se pueden obtener de la evaluación de las reglas de inferencia representan la versión de la entidad más adecuada para ser enviada al sitio de trabajo solicitante. En caso de que dicho sitio no soporte el procesamiento de la entidad asignada, entonces se envía un substituto. Los valores definidos para las versiones de las entidades y del substituto son:

- S – Substituto = 0
- V1 – Versión 1 = 1
- V2 – Versión 2 = 2
- V3 – Versión 3 = 3

Tomando en cuenta la **regla de producto**, el **total de posibilidades** y **resultados** sería: $3\ x\ 3\ x\ 3\ x\ 3\ x\ 3\ x\ 5 = 1215$, el primer 3 corresponde a las tres

opciones (bajo-medio-alto) de la velocidad del procesador; el segundo compete al espacio disponible en disco duro; el tercero concierne al tamaño de la RAM; el cuarto se refiere a la resolución horizontal de pantalla y el último 3 atañe a la entidad compartida; el 5 corresponde a las categorías MIME en que se dividen las entidades (texto, imagen, animación, audio y video).

Los campos utilizados en las tablas de verdad tienen el siguiente orden y funcionalidad: los primeros cuatro permiten comparar las capacidades del sitio participante, el quinto corresponde al tamaño de la entidad compartida y se confronta con los cuatro primeros. Los últimos cinco campos representan los resultados de las reglas de inferencia por categoría. A continuación se define lo que representa cada campo en las tablas:

1. **VP**: velocidad del procesador
2. **EDDD**: espacio disponible en el disco duro
3. **TRAM**: tamaño de la RAM
4. **RP**: resolución horizontal de pantalla
5. **TEC**: tamaño de la entidad compartida
6. **EDT**: entidad despachada en texto
7. **EDI**: entidad despachada en imagen
8. **EDA**: entidad despachada en animación
9. **EDAu**: entidad despachada en audio
10. **EDV**: entidad despachada en vídeo

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
B	B	B	B	B	V3	V3	V1	V1	V1
B	B	B	B	M	V2	V2	S	S	S
B	B	B	B	A	V1	V1	S	S	S
B	B	B	B	B	V3	V3	V1	V1	V1
B	B	B	B	M	V2	V2	S	S	S
B	B	B	B	A	V1	V1	S	S	S
B	B	B	A	B	V3	V3	V1	V1	V1
B	B	B	A	M	V2	V2	V1	S	S
B	B	B	A	A	V1	V1	S	S	S
B	B	M	B	B	V3	V3	V1	V1	V1
B	B	M	B	M	V2	V2	S	S	S
B	B	M	B	A	V1	V1	S	S	S
B	B	M	M	B	V3	V3	V1	V1	V1
B	B	M	M	M	V2	V2	V1	S	S
B	B	M	M	A	V1	V1	S	S	S
B	B	M	A	B	V3	V3	V1	V1	V1
B	B	M	A	M	V2	V2	V1	S	S
B	B	M	A	A	V1	V1	S	S	S
B	B	A	B	B	V3	V3	V1	V1	V1
B	B	A	B	M	V2	V2	S	V1	S
B	B	A	B	A	V1	V1	S	S	S
B	B	A	M	B	V3	V3	V1	V1	V1
B	B	A	M	M	V2	V2	V1	V1	S
B	B	A	M	A	V1	V1	S	S	S
B	B	A	A	B	V3	V3	V2	V1	V1
B	B	A	A	M	V2	V2	V1	V1	S
B	B	A	A	A	V1	V1	V1	S	S

Reglas de la 1 a la 27

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
B	M	B	B	B	V3	V3	V1	V1	V1
B	M	B	B	M	V2	V2	S	S	S
B	M	B	B	A	V1	V1	S	S	S
B	M	B	M	B	V3	V3	V1	V1	V1
B	M	B	M	M	V2	V2	V1	S	S
B	M	B	M	A	V1	V1	S	S	S
B	M	B	A	B	V3	V3	V1	V1	V1
B	M	B	A	M	V2	V2	V1	S	S
B	M	B	A	A	V1	V1	V1	S	S
B	M	M	B	B	V3	V3	V1	V1	V1
B	M	M	B	M	V2	V2	V1	S	S
B	M	M	B	A	V2	V1	V1	S	S
B	M	M	M	B	V3	V3	V2	V1	V1
B	M	M	M	M	V3	V3	V1	S	S
B	M	M	M	A	V2	V2	V1	S	S
B	M	M	A	B	V3	V3	V2	V1	V1
B	M	M	A	M	V3	V3	V1	S	S
B	M	M	A	A	V2	V2	V1	S	S
B	M	A	B	B	V3	V3	V1	V1	V1
B	M	A	B	M	V2	V2	V1	V1	S
B	M	A	B	A	V1	V1	V1	S	S
B	M	A	M	B	V3	V3	V2	V1	V1
B	M	A	M	M	V3	V3	V2	V1	S
B	M	A	M	A	V2	V2	V1	S	S
B	M	A	A	B	V3	V3	V2	V1	V1
B	M	A	A	M	V3	V3	V2	V1	S
B	M	A	A	A	V2	V2	V2	S	S

Reglas de la 28 a la 54

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
B	A	B	B	B	V3	V3	V1	V1	V1
B	A	B	B	M	V2	V2	S	S	S
B	A	B	B	A	V1	V1	S	S	S
B	A	B	M	B	V3	V3	V1	V1	V1
B	A	B	M	M	V2	V2	V1	S	S
B	A	B	M	A	V1	V1	S	S	S
B	A	B	A	B	V3	V3	V1	V1	V1
B	A	B	A	M	V2	V2	V1	S	S
B	A	B	A	A	V1	V1	V1	S	S
B	A	M	B	B	V3	V3	V1	V1	V1
B	A	M	B	M	V2	V2	V1	S	S
B	A	M	B	A	V1	V1	V1	S	S
B	A	M	M	B	V3	V3	V2	V1	V1
B	A	M	M	M	V3	V3	V2	S	S
B	A	M	M	A	V2	V2	V1	S	S
B	A	M	A	B	V3	V3	V2	V1	V1
B	A	M	A	M	V3	V3	V2	S	S
B	A	M	A	A	V2	V2	V1	S	S
B	A	A	B	B	V3	V3	V2	V1	V1
B	A	A	B	M	V2	V2	V1	V1	S
B	A	A	B	A	V1	V1	V1	S	S
B	A	A	M	B	V3	V3	V2	V1	V1
B	A	A	M	M	V3	V3	V2	V1	S
B	A	A	M	A	V2	V2	V1	S	S
B	A	A	A	B	V3	V3	V2	V1	V1
B	A	A	A	M	V3	V3	V2	V1	S
B	A	A	A	A	V2	V2	V2	S	S

Reglas de la 55 a la 81

VP	**EDDD**	**TRAM**	**RP**	**TEC**	**EDT**	**EDI**	**EDA**	**EDAu**	**EDV**
M	B	B	B	B	V3	V3	V1	V1	V1
M	B	B	B	M	V2	V2	S	S	S
M	B	B	B	A	V1	V1	S	S	S
M	B	B	M	B	V3	V3	V1	V1	V1
M	B	B	M	M	V2	V2	V1	S	S
M	B	B	M	A	V1	V1	S	S	S
M	B	B	A	B	V3	V3	V1	V1	V1
M	B	B	A	M	V2	V2	V1	S	S
M	B	B	A	A	V1	V1	S	S	S
M	B	M	B	B	V3	V3	V1	V2	V1
M	B	M	B	M	V2	V2	V1	V1	S
M	B	M	B	A	V1	V1	S	S	S
M	B	M	M	B	V3	V3	V2	V2	V1
M	B	M	M	M	V3	V3	V2	V1	S
M	B	M	M	A	V2	V2	V1	S	S
M	B	M	A	B	V3	V3	V2	V2	V1
M	B	M	A	M	V3	V3	V2	V1	S
M	B	M	A	A	V2	V2	V1	S	S
M	B	A	B	B	V3	V3	V2	V3	V1
M	B	A	B	M	V2	V2	V1	V2	S
M	B	A	B	A	V1	V1	S	V1	S
M	B	A	M	B	V3	V3	V2	V3	V1
M	B	A	M	M	V3	V3	V2	V2	S
M	B	A	M	A	V2	V2	V1	V1	S
M	B	A	A	B	V3	V3	V2	V3	V1
M	B	A	A	M	V3	V3	V2	V2	S
M	B	A	A	A	V2	V2	V1	V1	S

Reglas de la 82 a la 108

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
M	M	B	B	B	V3	V3	V1	V1	V1
M	M	B	B	M	V2	V2	V1	S	S
M	M	B	B	A	V1	V1	S	S	S
M	M	B	M	B	V3	V3	V2	V1	V1
M	M	B	M	M	V3	V2	V1	S	S
M	M	B	M	A	V2	V1	V1	S	S
M	M	B	A	B	V3	V3	V2	V1	V1
M	M	B	A	M	V3	V2	V2	S	S
M	M	B	A	A	V2	V1	V1	S	S
M	M	M	B	B	V3	V3	V1	V2	V1
M	M	M	B	M	V3	V2	V1	V1	S
M	M	M	B	A	V2	V1	V1	S	S
M	M	M	M	B	V3	V3	V2	V2	V3
M	M	M	M	M	V3	V3	V2	V1	V1
M	M	M	M	A	V3	V3	V1	S	S
M	M	M	A	B	V3	V3	V2	V2	V3
M	M	M	A	M	V3	V3	V2	V1	V1
M	M	M	A	A	V3	V3	V2	S	S
M	M	A	B	B	V3	V3	V2	V3	V1
M	M	A	B	M	V3	V2	V1	V3	S
M	M	A	B	A	V2	V1	V1	V2	S
M	M	A	M	B	V3	V3	V3	V3	V3
M	M	A	M	M	V3	V3	V2	V3	V1
M	M	A	M	A	V3	V3	V2	V2	S
M	M	A	A	B	V3	V3	V3	V3	V3
M	M	A	A	M	V3	V3	V3	V3	V2
M	M	A	A	A	V3	V3	V2	V2	V1

Reglas de la 109 a la 135

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
M	A	B	B	B	V3	V3	V1	V1	V1
M	A	B	B	M	V2	V2	S	S	S
M	A	B	B	A	V1	V1	S	S	S
M	A	B	M	B	V3	V3	V2	V1	V1
M	A	B	M	M	V3	V3	V1	S	S
M	A	B	M	A	V2	V2	V1	S	S
M	A	B	A	B	V3	V3	V2	V1	V1
M	A	B	A	M	V3	V3	V2	S	S
M	A	B	A	A	V2	V2	V1	S	S
M	A	M	B	B	V3	V3	V2	V2	V1
M	A	M	B	M	V3	V2	V1	V1	S
M	A	M	B	A	V2	V1	V1	S	S
M	A	M	M	B	V3	V3	V3	V2	V3
M	A	M	M	M	V3	V3	V2	V1	V1
M	A	M	M	A	V3	V3	V1	S	S
M	A	M	A	B	V3	V3	V3	V2	V3
M	A	M	A	M	V3	V3	V3	V1	V2
M	A	M	A	A	V3	V3	V2	S	V1
M	A	A	B	B	V3	V3	V2	V3	V1
M	A	A	B	M	V3	V2	V1	V3	S
M	A	A	B	A	V2	V1	V1	V2	S
M	A	A	M	B	V3	V3	V3	V3	V3
M	A	A	M	M	V3	V3	V3	V3	V2
M	A	A	M	A	V3	V3	V2	V2	V1
M	A	A	A	B	V3	V3	V3	V3	V3
M	A	A	A	M	V3	V3	V3	V3	V3
M	A	A	A	A	V3	V3	V3	V2	V2

Reglas de la regla 136 a la 162

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
A	B	B	B	B	V3	V3	V1	V1	V1
A	B	B	B	M	V2	V2	S	V1	S
A	B	B	B	A	V1	V1	S	S	S
A	B	B	M	B	V3	V3	V1	V1	V1
A	B	B	M	M	V2	V2	S	V1	S
A	B	B	M	A	V1	V1	S	S	S
A	B	B	A	B	V3	V3	V1	V1	V1
A	B	B	A	M	V2	V2	V1	V1	S
A	B	B	A	A	V1	V1	S	S	S
A	B	M	B	B	V3	V3	V1	V3	V1
A	B	M	B	M	V2	V2	V1	V2	S
A	B	M	B	A	V1	V1	S	V1	S
A	B	M	M	B	V3	V3	V2	V3	V1
A	B	M	M	M	V3	V3	V2	V2	S
A	B	M	M	A	V2	V2	V1	V1	S
A	B	M	A	B	V3	V3	V3	V3	V1
A	B	M	A	M	V3	V3	V2	V2	S
A	B	M	A	A	V2	V2	V2	V1	S
A	B	A	B	B	V3	V3	V2	V3	V1
A	B	A	B	M	V2	V2	V1	V3	S
A	B	A	B	A	V1	V1	V1	V2	S
A	B	A	M	B	V3	V3	V3	V3	V1
A	B	A	M	M	V3	V3	V2	V3	S
A	B	A	M	A	V2	V2	V2	V2	S
A	B	A	A	B	V3	V3	V3	V3	V1
A	B	A	A	M	V3	V3	V3	V3	S
A	B	A	A	A	V2	V2	V2	V2	S

Reglas de la regla 163 a la 189

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
A	M	B	B	B	V3	V3	V1	V1	V1
A	M	B	B	M	V2	V2	V1	V1	S
A	M	B	B	A	V1	V1	S	S	S
A	M	B	M	B	V3	V3	V2	V1	V1
A	M	B	M	M	V3	V3	V1	V1	S
A	M	B	M	A	V2	V2	V1	S	S
A	M	B	A	B	V3	V3	V2	V1	V1
A	M	B	A	M	V3	V2	V2	V1	S
A	M	B	A	A	V2	V1	V1	S	S
A	M	M	B	B	V3	V3	V2	V3	V1
A	M	M	B	M	V3	V2	V1	V3	S
A	M	M	B	A	V2	V1	V1	V2	S
A	M	M	M	B	V3	V3	V3	V3	V3
A	M	M	M	M	V3	V3	V2	V3	V1
A	M	M	M	A	V3	V3	V2	V2	S
A	M	M	A	B	V3	V3	V3	V3	V3
A	M	M	A	M	V3	V3	V3	V3	V2
A	M	M	A	A	V3	V3	V3	V2	V1
A	M	A	B	B	V3	V3	V2	V3	V1
A	M	A	B	M	V3	V2	V2	V3	S
A	M	A	B	A	V2	V1	V1	V3	S
A	M	A	M	B	V3	V3	V3	V3	V3
A	M	A	M	M	V3	V3	V3	V3	V2
A	M	A	M	A	V3	V3	V2	V3	V1
A	M	A	A	B	V3	V3	V3	V3	V3
A	M	A	A	M	V3	V3	V3	V3	V3
A	M	A	A	A	V3	V3	V3	V3	V2

Reglas de la regla 190 a la 216

VP	EDDD	TRAM	RP	TEC	EDT	EDI	EDA	EDAu	EDV
A	A	B	B	B	V3	V3	V1	V1	V1
A	A	B	B	M	V2	V2	S	V1	S
A	A	B	B	A	V1	V1	S	S	S
A	A	B	M	B	V3	V3	V2	V1	V1
A	A	B	M	M	V3	V3	V1	V1	S
A	A	B	M	A	V2	V2	S	S	S
A	A	B	A	B	V3	V3	V2	V1	V1
A	A	B	A	M	V3	V3	V2	V1	S
A	A	B	A	A	V2	V2	V1	S	S
A	A	M	B	B	V3	V3	V2	V3	V1
A	A	M	B	M	V3	V2	V1	V3	S
A	A	M	B	A	V2	V1	V1	V2	S
A	A	M	M	B	V3	V3	V3	V3	V3
A	A	M	M	M	V3	V3	V2	V3	V2
A	A	M	M	A	V3	V3	V1	V2	V1
A	A	M	A	B	V3	V3	V3	V3	V3
A	A	M	A	M	V3	V3	V3	V3	V3
A	A	M	A	A	V3	V3	V2	V2	V2
A	A	A	B	B	V3	V3	V2	V3	V1
A	A	A	B	M	V3	V2	V1	V3	S
A	A	A	B	A	V2	V1	S	V3	S
A	A	A	M	B	V3	V3	V3	V3	V3
A	A	A	M	M	V3	V3	V3	V3	V3
A	A	A	M	A	V3	V3	V2	V3	V2
A	A	A	A	B	V3	V3	V3	V3	V3
A	A	A	A	M	V3	V3	V3	V3	V3
A	A	A	A	A	V3	V3	V3	V3	V3

Reglas de la 217 a la 243

Bibliografía

[Alonso, 2007] Sergio A. Alonso, *Redes Libres: Técnicas para armado de redes LAN utilizando software libre*, Primera edición, Argentina, 2007.

[Appelt, 2001] W. Appelt, *"What Groupware Functionality Do Users Really use? Analysis of the Usage of the BSCW System"*, In Proc. of PDP'2001, the 9th Euromicro Workshop on Parallel and Distributed Processing, pp. 337-341, IEEE Computer Society Press, Mantova (Italia), 7-9 Febrero 2001.

[Arora et al., 2002] A. Arora, C. Haywood and K.S. Pabla, *JXTA for J2ME: Extending the Reach of Wireless With JXTA Technology*, Technical Report 650 960-1300, Sun Microsystems, Inc, pp. 1-4, Palo Alto, California (Estados Unidos de América), Marzo 2002.

[Beato y Franco, 1998] M. Beato y J. Franco, *Manual Avanzado de Windows NT Server 4.0,* primera edición, Anaya, España, 1998.

[Bentley et al., 1997] R. Bentley, T. Horstmann and J. Trevor, *"The World Wide Web as enabbling technology for CSCW*: The case of BSCW", Computer Supported Cooperative Work, vol. 6, num. 2/3, pp. 111-134, 1997.

[Berzal, 2002] F. Berzal, *ART: Un método alternativo para la construcción de árboles de decisión*, Tesis Doctoral, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada E. T. S. Ingeniería Informática, Junio 2002.

[Chahuara, 2005] J. C. Chahuara, *Control neuro-difuso aplicado a una Grúa Torre*, Ingeniero Electrónico, Facultad de Ingeniería Eléctrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, 2005.

[Chung and Dewan, 2004] G. Chung, y P. Dewan, *Towards dynamic collaboration architectures, Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'2004)*, ACM Press, pp. 1-10, Chicago, IL (Estados Unidos de América), 2004.

[Coulouris et al., 2001] G. Coulouris, J. Dollimore and T. Kindberg, *Distributed Systems: Concepts and Design*, third edition, Pearson Education, China, 2001.

[Decouchant et al., 2001] D. Decouchant, J. Favela and A. M. Martínez Enríquez, *PIÑAS: A Middleware for Web Distributed Cooperative Authoring*, In Proceedings of the 2001 Symposium on Applications and the Internet (SAINT'2001), IEEE Computer Society and IPJ Information Processing Society of Japan, pp. 187-194, San Diego, CA (Estados Unidos de América), 2001.

[Dewan and Choudhary, 1992] P. Dewan and R. Choudhary, *A High-Level and Flexible Framework for Implementing Multiuser Interfaces, ACM Transactions on Information Systems*, Vol. 10, No. 4, Octubre 1992, 345-380

[Dourish, 1998] P. Dourish, *Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications*, ACM Transactions on Computer-Human Interaction, 5(2): pp. 109-155, 1998.

[Goland et al., 1999] Y.Y.Goland, E.J. Whitehead Jr., a Faizi, S. R. Carter and D. Jensen, *HTTP Extensions for Distributed Authoring - WebDAV*, RFC 2518, W3C Standard, Febrero 1999.

[Hill et al., 1993] R. D. Hill, T. Brinck, J. F. Patterson, S. L. Rohall and W. T. Wilner, *Rendezvous Languaje Communications of the ACM*, Vol. 36, No. 1, Enero 1993, 62-67.

[Ibrahim, 2004] A. M. Ibrahim, *Fuzzy Logic for Embedded Systems Applications*, Primera edición, Elsevier, Estados Unidos de América, 2004.

[Lukosch, 2003] S. Lukosch, *Transparent and Flexible Data Sharing for Synchronous Groupware*, Tesis de Doctorado, FernUniversität, Hagen (Alemania), Mayo 2003.

[Mateu, 2004] C. Mateu, *Software libre: Desarrollo de aplicaciones Web*, Primera edición, Eureca Media, España, 2004.

[Mendoza, 2006] S.G. Mendoza, *Gestion d´entités partagées dans un environnement de production coopérative Web*, PhD. Thesis, Laboratorie LSR-IMAG, Institut National Polytechnique de Grenoble, septiembre 2006.

[Millan, 2006] R. J. Millán Tejedor, *Domine las redes P2P: "Peer to Peer.ºrígenes, funcionamiento y legislación del P2P, selección y configuración del acceso de banda ancha a Internet*, Primera edición, Creaciones Copyright, España, 2006.

[Micha, 1998] E. Micha, *Matemáticas discretas*, Primera edición, LIMUSA Noriega Editores, México, 1998.

[Mohler, 1998] J. L. Mohler, *Aprendiendo a convertirse en Webmaster*, primera edición en español, Prentice Hall, Méxco, 1998.

[Morán et al., 2002] A. L. Morán, D. Decouchant, J. Favela, A. M. Martínez Enríquez, B. González Beltrán and S. Mendoza, *PIÑAS: Supporting a Community of Co-Authors on the Web*, In Proceedings of the Fourth International Conference on Distributed Communities on the Web (DCW'2002), Springer Verlag, LNCS 2468, pp. 114-125, Sydney (Australia), 3-5 Abril 2002.

[NCSA] NCSA, NCSA Habanero Homepage, http://www.ncsa.uiuc.edu/SDG/Software /Habanero/HabaneroHome.html.

[O'Grady, 1996] T. O'Grady, *Flexible Data Sharing in a Groupware Toolkit*, Tesis de Maestría, Department of Computer Science, University of Calgary, Calgary, Alberta (Canadá), Noviembre 1996.

[Patterson et al., 1996] J. F. Patterson, M. Day and J. Kucan, *Notification Servers for Synchronous Groupware, in Proceedings of the ACM Conference on Computer Supported Cooperative Work*, ACM Press, Noviembre 1996, 122-129.

[Phillips, 1999] W.G. Phillips, *Architectures for Synchronous Groupware*, Technical Report 1999-425, Department of Computing and Information Science, Queen´s University, Kingston, ON (Canadá), Mayo 1999.

[Pressman, 2002] R. S. Pressman, *Ingeniería de software: Un enfoque práctico,* traducido de la quinta edición en inglés, McGraw-Hill, España, 2002.

[Ramduny and Dix, 1997] D. Ramduny and A. Dix, *Why, What, Where, When: Architectures for Cooperative Work on the WWW*, In Proceedings of the Conference on Human-Computer Interaction (HCI'97), Eds. H. Thimbleby, B. O'Connaill and P. Thomas, Springer, pp. 283-301, Bristol (Reino Unido), 1997.

[Roseman and Greenberg , 1996] M. Roseman and S. Greenberg, *Building Real-Time Groupware with GroupKit, A Groupware Toolkit, ACM Transactions on Computer-Human Interaction*, Vol. 3, No. 1, Marzo 1996, 66-106.

[Roth, 2004] J. Roth, *The Resource Framework for Mobile Applications: Enabling Collaboration Between Mobile Users*, Computer Science Department, University of Hagen, Hagen (Alemania), 2004.

[Roth and Unger, 2000] J. Roth and C. Unger, *An extensible classification model for distribution architectures of synchronous groupware*, In Proceedings of the 4th International Conference on the Design of Cooperative Systems (COOP'2000), IOS Press, Sophia Antipolis (Francia), Mayo 2000.

[Rowe, 1996] J. Rowe, *Creación de servidores de bases de datos para Internet con CGI*, Primera edición en español, Prentice-Hall, México, 1996.

[Raya y Raya, 2002] J. L. Raya y C. Raya, *Redes Locales*, primera edición, Alfaomega, México, 2002.

[Simao et al., 1997] J. Simao, H. J. Domingos, J. L. Martins and N. Preguica, *Supporting Synchronous Groupware with Peer Object-Groups,* In Proceedings of the Third USENIX Conference on Object-Oriented

Technologies (COOTS). Portland, Oregon, USA, June 1997.

[Streitz et al., 1994] N. A. Streitz, J. GeiBler, J. M. Haake and J. Hol, *DOLPHIN: Integrated Meeting Support across LiveBoards, Local and Remote Desktop Environments,* In Proceedings of the ACM '94 Conference on Computer Supported Cooperative Work (CSCW '94), Chapel Hill, North Carolina, Okt. 22-26, 1994, 345-358.

[Tanenbuam and Van Steen, 2002] A. S. Tanenbuam and M. Van Steen, *Distributed Systems Principles and Paradigms*, Prentice Hall, Estados Unidos de America, 2002.

[Zadeh, 1965] L. A. Zadeh, *Fuzzy Sets*, Department of Electrical Engineering and Electronics Research Laboratory, University of California: Berkeley, California (Estados Unidos de América), 1965.