OBJETOS MÓVILES EN UN SISTEMA DISTRIBUIDO SOBRE EL WWW

Ricardo De la Rosa
María Consuelo Franky
Pablo Figueroa
e-mails : {ri-de.cfranky,pfiguero}@uniandes.edu.co
Universidad de los Andes, Bogotá-Colombia

Carlos Varela
e-mail : cvarela@ncsa.uiuc.edu
University of Illinois, Urbana-USA

Resumen

En un sistema distribuido, de área mundial, que tiene objetos que ofrecen una interfaz de servicios RPC, la técnica de cadenas SSP [SDP92] permite que puedan migrar a través de la red, transparentemente para sus clientes. Sin embargo, esta técnica representa las referencias a objetos por medio de cadenas SSP extendidas sobre una secuencia de computadores arbitrariamente larga, y esto tiene dos inconvenientes : Primero, la disponibilidad de los objetos referenciados puede depender de que numerosos sitios no fallen. Segundo, la recolección de basura distribuida es compleja. Respecto a recuperación ante fallas, la técnica de cadenas SSP propone buscar exhaustivamente en el sistema, lo que puede no ser realizable, en un sistema con innumerables sitios. Lo que proponemos en este artículo son cambios a esta técnica para no dejar alargar las cadenas SSP y no hacer búsquedas globales.

Palabras claves. Sistemas distribuidos, objetos móviles, agentes móviles, objetos distribuidos, objetos en el WWW, aplicaciones migrantes.

1 Introducción

El contexto de este estudio es un sistema distribuido de área mundial (e.g. el WWW) que tiene recursos representados como objetos remotos que ofrecen un conjunto de servicios vía una interfaz pública. Esta interfaz se puede acceder usando un mecanismo de invocación remota de métodos (e.g. Java RMI [Sun97]).
Los objetos de interés tienen un *esquema de representación* de *una-sola-copia*. Es decir, el estado y el comportamiento de cada objeto conforman una unidad localizada en un solo sitio[1] del sistema [BAB96]. En otras palabras, no tratamos ni con objetos replicados ni con objetos fragmentados [MGLS94].

El proyecto descrito en este artículo busca establecer los mecanismos y protocolos apropiados para tener *objetos móviles* en el contexto mencionado. Un objeto móvil (i.e. OM) es una identidad que puede moverse por varios sitios del sistema. Cuando un objeto migra de un sitio a otro, su estado antes y después de la migracion permanece igual [SDP92]. Además, es deseable que la migración de un objeto sea transparente para las entidades clientes del objeto.

Concretamente, la pregunta que queremos resolver en nuestro proyecto es : Cómo identificar, referenciar y acceder objetos móviles, de manera apropiada, en una infraestructura distribuida de área mundial ?

[1] En este contexto, sitio es sinónimo de espacio de direcciones.

Analizando esta pregunta se pueden identificar cinco subproblemas :

- Movilidad del objeto y validez de las referencias existentes.
- Paso de referencias entre dos sitios.
- Recolección de basura distribuida, donde un objeto es basura si ningún otro objeto lo está referenciando. No consideramos el caso de la basura cíclica en el cual dos objetos que solo se referencian entre si son considerados basura.
- Servicio de nombres que soporte objetos móviles. La principal exigencia a este servicio es que, el nombre de un objeto sea independiente de su localización física.
- Debido a que estos objetos móviles se consideran en el contexto de un sistema distribuido de área mundial. Entonces, es obligación tener mecanismos y protocolos que manejen tolerancia a fallas.

Para resolver los problemas enunciados nos vamos a basar en la técnica de Cadenas SSP presentada en [SDP92]. Nuestro aporte está en proponer mejoras a esta técnica, y estudiar la utilidad práctica de estas ideas para mejorar la tecnología actual del WWW.

Respecto a la estructura de este documento: En la sección 2 mencionamos las motivaciones de este trabajo. La sección 3 describe la técnica de cadenas SSP incluyendo nuestras mejoras. La técnica original y la técnica propuesta (i.e. con nuestras mejoras) son comparadas en la sección 4. En la sección 5 presentamos una aplicación práctica de los objetos móviles. En la sección 6 mencionamos el trabajo relacionado con este proyecto. Finalmente, la sección 7 presenta las conclusiones.

2 Motivaciones para utilizar objetos móviles

En general, tener objetos móviles (i.e. OMs) constituye un esquema muy flexible para construir sistemas distribuidos en el WWW. Ejemplos en los que resulta útil tener OMs son :

- Un problema serio del WWW es la ausencia de integridad referencial en los enlaces entre los documentos HTML. Su principal causa es que los administradores de los hipertextos no los pueden mover de manera que las referencias que los apuntan sigan siendo válidas. Si se modelaran estos documentos como OMs, los documentos podrían migrar y los enlaces se mantendrían consistentes.
- Hay identidades de la realidad que periódicamente se desplazan de un lugar geográfico a otro. Además, existen sistemas que manejan la información de estas identidades. Por razones de flexibilidad y eficiencia (en el acceso a la información) resulta conveniente modelar estas identidades como OMs. Así, cuando la identidad se desplaze, el OM asociado migrará al nodo del sistema correspondiente a la ubicación del objeto. Por ejemplo, supongamos que una empresa traslada uno de sus trabajadores de la ciudad A a la ciudad B. Sería interesante que el objeto que tiene la información del trabajador fuera capaz de migrar del computador de la sucursal en A al "host" de la sucursal en B.
- Antes de mirar esta motivación es importante decir que : el concepto de *agente móvil* [BC96] es un caso particular del concepto de objeto móvil.
  Recientemente surgió un nuevo genero de aplicaciones con interfaz al usuario : las *aplicaciones migratorias* pueden moverse de un computador a otro, manteniendo intacto el estado de sus interfaces al usuario [BC96]. La migración de aplicaciones puede resultar útil para numerosos tipos de aplicaciones colaborativas basadas en agentes móviles. Por ejemplo, la metáfora del correo interactivo en la que los mensajes son capaces de interactuar inteligentemente con los usuarios. Además, el receptor del mensaje puede reenviarlo después de haber interactuado con él.
- Empleando objetos móviles se han diseñado interesantes arquitecturas distribuidas. Usando estas arquitecturas en la implementación de aplicaciones distribuidas se pueden resolver naturalmente numerosos problemas, a nivel de aplicación, de la computación móvil [BP96].

3 Mecanismos y protocolos para soportar objetos móviles

El objetivo de esta sección es describir los mecanismos y protocolos necesarios para tener OMs usando referencias representadas por cadenas SSP [SDP92]. Simultaneamente, incluiremos las mejoras que proponemos a esta técnica.

3.1 Conceptos básicos del esquema de cadenas SSP

Para comenzar, expliquemos la notación usada en las figuras : Un cuadrado grande representa un sitio (i.e. un computador) del sistema. Los sitios los identificamos con un número en la parte superior (e.g. en la figura 1 tenemos los sitios 1, 2 y 3). Sin embargo, en la realidad no necesitamos tener los sitios numerados u ordenados. Un círculo esquematiza un objeto móvil o no móvil. La letra minúscula que está en su interior es el identificador del objeto (e.g. en la figura 1 tenemos los objetos x, n y t). Un cuadrado pequeño identificado al interior con R_i corresponde a la raiz local del sitio i. Esta raiz representa al servidor que crea y exporta los objetos (e.g. un servidor Java RMI [Sun97] o un ORB en la arquitectura CORBA [OMG95]).

Un pentágono convexo identificado con o_i corresponde al representante del objeto o en el sitio i. Un objeto tiene a lo sumo un representante en cada sitio. Un pentágono cóncavo etiquetado con o^i corresponde a una cavidad en el sitio i, que pertenece a una cadena cuyo destino es el objeto o. Como pueden haber varias cavidades pertenecientes a cadenas que desembocan en un mismo objeto, vamos a diferenciar estas cavidades usando el sistema primado (e.g. en la figura 2 tenemos las cavidades t^3, t'^3 y t''^3).

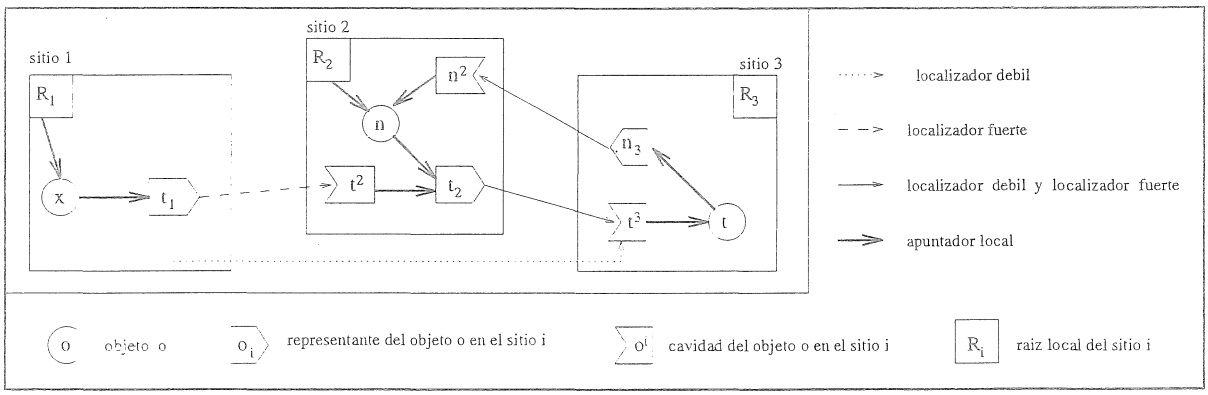

Figura 1: Una referencia está representada por una cadena de parejas Representante-Cavidad.

En el esquema de cadenas SSP, una referencia a un objeto móvil es representada por una cadena de parejas, de objetos auxiliares, Representante-Cavidad (i.e. una cadena SSP[2] [SDP92]). En la figura 1 hay tres referencias : una de x a t, otra de n a t, y otra de t a n. Además, hay referencias directas que están compuestas de una pareja representante-cavidad (e.g. la cadena $n \rightarrow t_2 \rightarrow t^3 \rightarrow t$). También hay referencias indirectas que están compuestas de más de una pareja representante-cavidad (e.g. la cadena $x \rightarrow t_1 \rightarrow t^2 \rightarrow t_2 \rightarrow t^3 \rightarrow t$).

En la técnica original, las cadenas SSP pueden tener longitud arbitraria. En el esquema que proponemos, típicamente, las cadenas están compuestas de una SSP (i.e. una pareja representante-cavidad); en el peor de los casos llegan a estar compuestas de tres SSPs.

Adicionalmente, cada representante contiene dos componentes (i.e. dos apuntadores) : un *localizador fuerte* y un *localizador débil*.

El localizador fuerte referencia la siguiente cavidad en la cadena. El localizador débil puede apuntar a una cavidad más cercana al objeto destino, comparada con la referenciada por el localizador fuerte. En la figura 1, el localizador fuerte del representante t_1 apunta a la cavidad t^2, mientras que el localizador débil apunta a la cavidad t^3 que se encuentra más cerca al objeto t. Hay ocasiones en las que el localizador débil y el fuerte apuntan la misma cavidad. En la figura 1, tanto el localizador fuerte como el débil de t_2 apuntan a t^3.

[2]SSP : En inglés, Stub-Scion Pairs; en francés, Paires Souche-Scion.

3.2 Paso de un sitio a otro de las referencias que apuntan a un objeto

Una de las características deseables, en este proyecto, es permitir las siguientes dos situaciones :

- Supongamos que un objeto tiene una referencia a un segundo objeto que es móvil. El primero le puede invocar un método a un tercer objeto y pasarle esta referencia como argumento de entrada.
- De nuevo, un primer objeto tiene una referencia a un segundo objeto que es móvil. Un tercer objeto puede invocarle un método al primero cuyo valor de retorno sea la referencia al segundo.

En ambas situaciones, el primer y el tercer objeto pueden ser migrantes o no migrantes. Los tres objetos pueden localizarse en diferentes sitios.

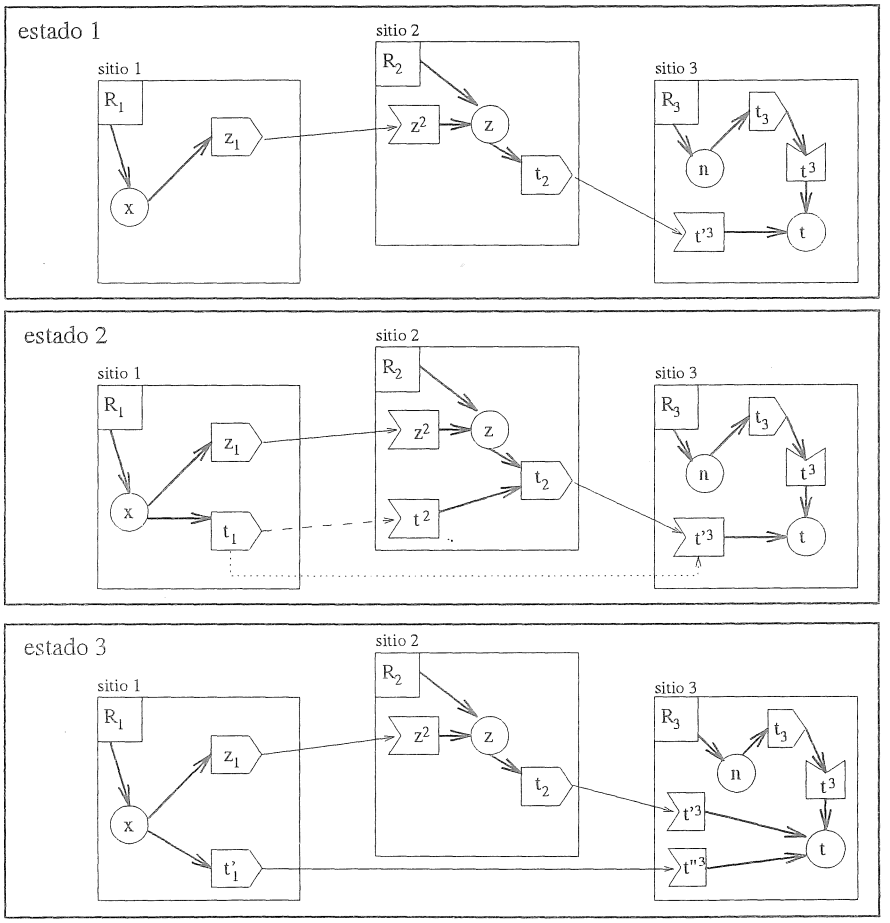

Figura 2: Paso de una referencia apuntando a t, desde el sitio 2 al sitio 1

Vamos a explicar el protocolo mediante un ejemplo. En la figura 2 (estado 1), el objeto x invoca un método de z, el cual retorna una referencia a t. Para lograr esto, se crea la cavidad t^2, la cual se pone a referenciar al representante de t en el sitio 2 (i.e. t_2). Luego, en el mensaje que z le manda los argumentos de salida y el valor de retorno a x, nuestro ambiente en el sitio 2 empaqueta el apuntador a la cavidad t^2 y el apuntador a la cavidad t'^3 (esta última referencia se consigue con la información que tiene el representante t_2). Cuando este mensaje llega al sitio 1 se crea el representante de t en 1 (i.e. t_1), a su localizador fuerte se le asigna el apuntador a la cavidad t^2, y al localizador débil se le asigna el apuntador a la cavidad t'^3. Después, al objeto x se lo pone a referenciar a t_1, y así x logra obtener una referencia a t compuesta de una cadena SSP indirecta

(figura 2 - estado 2).

Es importante decir que las cadenas SSP indirectas son indeseables por dos razones : Primero, por la falta de disponibilidad si un sitio se cae, y por este nodo pasaba una parte de la cadena SSP. Segundo, por desempeño, ya que los mensajes de las invocaciones remotas tienen que llegar al objeto destino, y muchas veces tienen que recorrer estas cadenas. Adicionalmente, hay que mencionar que, después de una invocacion remota a un OM se obtiene una referencia directa entre el invocador y el OM (§3.5).

Por lo anterior, nosotros proponemos que apenas un objeto reciba una referencia a un OM (i.e. objeto móvil), él la utilice para invocar un método nulo. Esta invocación no tiene ningún efecto sobre el receptor pero sirve para establecer una referencia directa entre el invocador y el OM (i.e. el receptor). En la figura 2 (estado 2), x invoca el método nulo sobre t, y así x obtiene una referencia directa a t (figura 2 - estado 3).

3.3 Servicio de nombres que soporta objetos móviles

La manera en la que se manejan los nombres en nuestro esquema es similar a la forma en que lo hace Java RMI [Sun97]. Primero, se utiliza el mecanismo de URLs para nombrar los objetos. Segundo, los objetos registran su nombre en su sitio de nacimiento. El nombre de un objeto es el URL formado por la concatenación del nombre del sitio de nacimiento y un identificador del objeto (interno al sitio) que tiene semántica para el propietario del objeto. Por ejemplo, supongamos que en el sitio "isis.uniandes.edu.co" nació un objeto cuyo identificador interno es "HelloServer", entonces el nombre de este objeto es el URL : "//isis.uniandes.edu.co/HelloServer".

El objeto registra su nombre con el servidor de nombres (representado como un objeto no migrante) de su sitio. El efecto de esta acción es que este servidor crea una referencia directa hacia el objeto registrado, y en una tabla asocia el nombre del objeto a esta referencia. De otro lado, también se crea una referencia directa del objeto móvil hacia su servidor de nombres (figura 3). Esta referencia se usa para que el objeto le informe a este servidor, cada vez que él haya migrado. Así el servidor de nombres es el primero en obtener una referencia directa a el objeto móvil. La figura 3 ilustra el funcionamiento del servicio de nombres : Aquí el objeto n es el servidor de nombres del sitio 3.

Cuando un objeto o aplicación cliente desea obtener una referencia a un objeto móvil, a partir de su nombre, le hace esta petición al servidor de nombres que tiene registrado el objeto deseado[3] (figura 3). Este servidor lo único que hace es pasarle una referencia del objeto al cliente siguiendo el protocolo de paso de referencias (sección 3.2). Así, el cliente obtiene una referencia directa al objeto deseado. En la figura 3, el objeto móvil t está registrado con el servidor de nombres n (n es un objeto no migrante). En el estado 1 de esta figura, z le solicita a n una referencia a t. En el estado 2, n le pasa a z una referencia indirecta a t. En el estado 3, z obtiene una referencia directa a t mediante una invocación nula a este objeto, usando la referencia indirecta.

3.4 Migración de un objeto de un sitio a otro

Una diferencia importante con el trabajo descrito en [SDP92] es que en nuestro esquema, periódicamente, cada entidad que tiene una referencia a un objeto móvil debe enviarle un mensaje a este OM. El objetivo de este mensaje es que el OM registre que esta referencia está siendo usada. Si este mensaje no es recibido después de un lapso de tiempo, el objeto móvil asume que una referencia menos lo está apuntando.

En la sección 3.5 vamos a ver que un segundo objetivo de estas invocaciones periódicas, es establecer una cadena directa entre el (objeto o aplicación) cliente y el objeto móvil. Ahora, supongamos que todos los clientes se reportan al objeto cada T unidades de tiempo. Entonces, si el objeto no migra durante un intervalo que dura T, al final de este lapso todas las referencias a este objeto serán representadas como cadenas directas.

[3] Este servidor está localizado en el sitio de nacimiento del objeto migrante, y el nombre de este sitio hace parte del nombre del objeto.

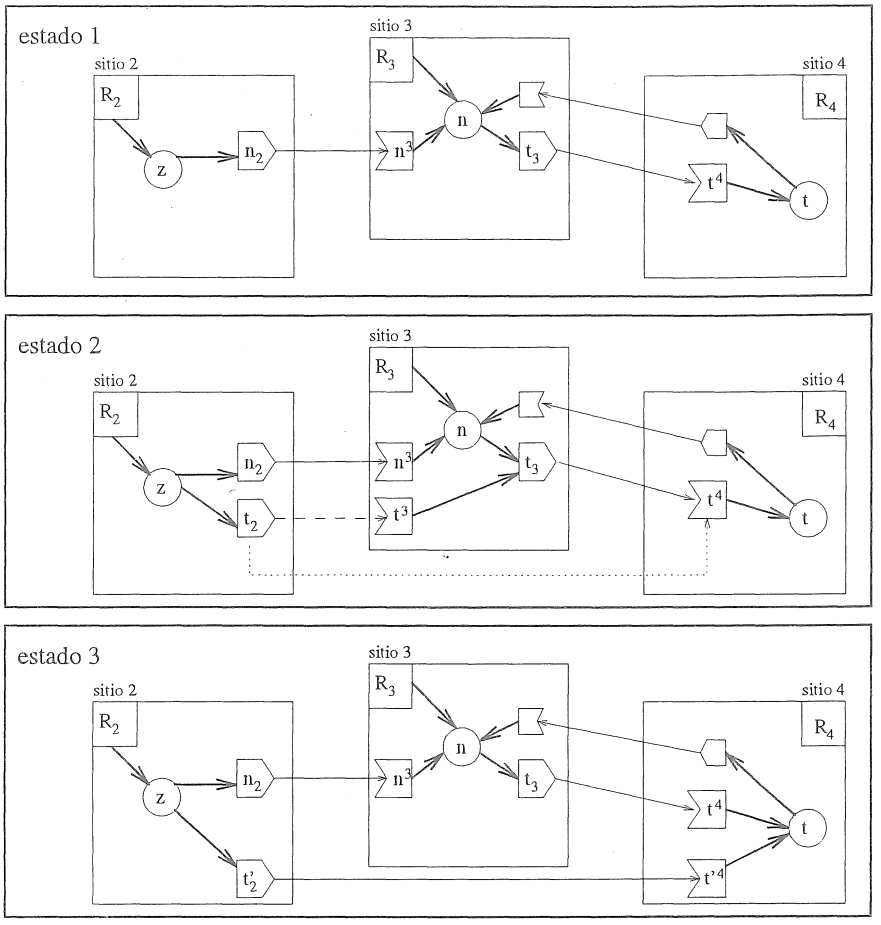

Figura 3: Servicio de nombres para objetos móviles.

Cuando a un objeto móvil le invocan el método de migrar se sigue un protocolo de migración. En la figura 4 (estado 1) el objeto t recibe de z una petición de migrar del sitio 3 al 4.

Protocolo de Migración

1. El objeto les ordena a las cavidades que lo referencian que represen todos los mensajes de aplicación correspondientes a invocaciones remotas. Estos mensajes serán almacenados para ser procesados después de que el objeto haya migrado. Los mensajes de control correspondientes a invocaciones cuyo objetivo es acortar las cadenas siguen siendo procesados normalmente.

2. El objeto espera un periodo T para lograr que todas las referencias a él sean directas.

3. El objeto les informa a las cavidades que lo referencian, que retengan todos los mensajes de control.

4. El objeto espera a que finalicen las ejecuciones en curso de los métodos. En ese momento, no hay manera de modificar el estado interno del objeto. Este hecho se muestra en la figura 4 (estado 2) representando el objeto con doble círculo.

5. Se crea una copia del OM en el sitio destino. En la figura 4 (estado 2), se crea el objeto t' en el sitio 4. Luego se envía al sitio 4 el estado de t, y se le asigna a t'.

6. Se establece una nueva pareja SSP donde la cavidad apunta al nuevo objeto y está en el sitio destino de la migración. El representante se crea en el sitio donde está el objeto original, y tanto su localizador débil

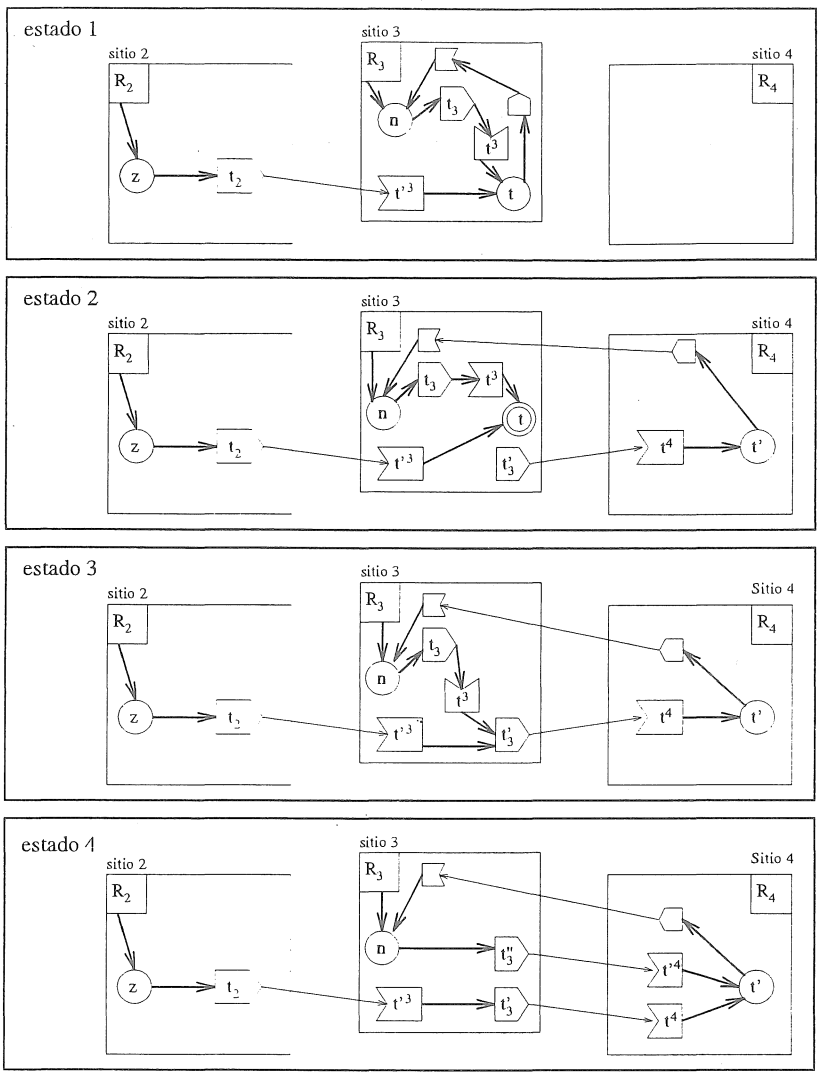

Figura 4: Migración del objeto t del sitio 3 al sitio 4.

como el fuerte apuntan a la nueva cavidad. En la figura 4 (estado 2), la nueva pareja está compuesta por el representante t'_3 y la cavidad t^4.

7. Los apuntadores locales que referencian al objeto original se ponen a apuntar al nuevo representante. En la figura 4 (estados 2 y 3), los apuntadores de las cavidades t^3 y t'^3 que apuntaban a t comienzan a apuntar a t'_3.

8. Se destruye el objeto original. En la figura 4 (estados 2 y 3), se elimina t.

9. El nuevo objeto le informa a su servidor de nombres (usando la referencia directa que todo objeto tiene a su correspondiente servidor) que él ya está localizado en el sitio destino de la migración. Gracias a esta señal, este servidor le hace al objeto una invocación nula (usando una referencia indirecta), con el objetivo de obtener una referencia directa. Este servidor se queda bloqueado hasta obtener esta referencia directa. En la figura 4 (estado 3), el objeto t' le informa a su servidor de nombres n que él ya está en el sitio 4. En ese momento n le hace una invocación nula a t' usando una referencia indirecta (i.e. la cadena $n \rightarrow t_3 \rightarrow t^3 \rightarrow t'_3 \rightarrow t^4 \rightarrow t'$). En la figura 4 (estado 4), el objeto n logra tener una referencia directa a t'.

10. Se valida la migración, y esto se hace ordenándole a las cavidades que dejen de represar los mensajes y les permitan fluir por la cadenas SSP que desemboca en el objeto que migró. En la figura 4 (estado 3), la cavidades t^3 y t'^3 dejan proseguir los mensajes que tenían almacenados para ser procesados. El objeto t' procesa los mensajes de aplicación y nuestro ambiente procesa los mensajes de control. Uno de estos mensajes de control es el enviado al final del paso anterior para que el servidor n obtenga una referencia directa a t'.

3.5 Invocación de métodos remotos

El principal efecto de una invocación de un método sobre un objeto móvil es recortar la cadena SSP que representa la referencia utilizada en este llamado.
La política detrás de nuestros mecanismos es, en lo posible, tener referencias directas (i.e. compuesta de una pareja SSP), lo cual implica ser agresivos en el recorte de cadenas. Por esta razón, la cadena más larga que se podría llegar a tener estaría compuesta por 3 parejas SSP. A continuación, describimos cómo se harían las invocaciones en nuestro sistema, de acuerdo a los tipos de referencias que se generan :

a. **Referencia directa del cliente hacia el objeto móvil**

Este es el caso de la referencia que n tiene hacia t, en la figura 1. Después de la invocación, un "stub" tipo RPC que hace parte del cliente empaqueta en un mensaje el código de operación del método que se está invocando, y los argumentos del llamado. Este mensaje es enviado al representante local del objeto destino, el cual lo reenvía a su cavidad correspondiente. Después, la cavidad le envía el mensaje al objeto destino. El objeto desempaqueta el código de operación y los argumentos de ésta invocación. Se realiza la operación respectiva y los valores de retorno son empaquetados en un nuevo mensaje. Este mensaje es enviado diréctamente al objeto o a la aplicación cliente.

b. **Referencia indirecta y además el localizador débil apunta a la cavidad (de la cadena SSP) más cercana al objeto**

Un localizador débil que cumpla esta característica se dice que es un localizador exacto, de lo contrario, se dice que es un localizador inexacto. Este es el caso de la referencia que x tiene hacia t en la figura 2 (estado 2). En este diagrama, x hace una invocación nula a t cuyo único objetivo es acortar esta referencia indirecta. La cadena directa obtenida puede verse en el estado 3 de esta misma figura.
Respecto a la invocación, el mensaje con la información de los argumentos y el código de operación del método, es enviado usando el localizador débil. La cavidad que lo recibe lo reenvía al objeto destino. Este objeto ejecuta el método y empaqueta los resultados en un segundo mensaje, llamémoslo m'. Adicionalmente, la plataforma se da cuenta que el mensaje llegó por medio de un localizador débil, lo cual quiere decir que la referencia que se utilizó es indirecta. Por tal razón, se crea una nueva cavidad (en la figura 2 - estado 3, esta cavidad es t''^3) para una nueva referencia directa. El apuntador a esta cavidad también es empaquetado en el mensaje m'. Luego, m' es enviado diréctamente al objeto o a la aplicación cliente.
En la recepción de m', la plataforma se da cuenta que en el mensaje viene un apuntador a una cavidad nueva y un flag que indica que se debe establecer una referencia directa utilizando esta cavidad. Entonces, se crea un representante en el sitio del cliente (en la figura 2 - estado 3, este representante es t'_1), y lo ponemos a apuntar a la nueva cavidad. El apuntador local que sale del objeto o la aplicación cliente, lo hacemos apuntar al nuevo representante. Finalmente, se elimina la pareja SSP que se volvió basura. En la figura 2 (estado 2), esta pareja es el representante t_1 y la cavidad t^2.

c. **Referencia indirecta y además el localizador débil es inexacto**

Un ejemplo de este caso se puede ver en el estado 4 de la figura 4, y es la referencia que z tiene hacia t'. Para la invocación se procede de la siguiente manera :

1. El mensaje con la información de los argumentos y el código de operación del método se envía utilizando el localizador débil. En la figura 4 (estado 4), el mensaje llegaría a la cavidad t'^3.

2. La cavidad que recibe este mensaje lo reenvía al siguiente representante que está en la cadena fuerte, el cual se lo entrega a su cavidad correspondiente. En nuestro esquema, esta cavidad obligatoriamente está en el mismo sitio que el objeto destino[4].
3. En el análisis del mensaje que hace esta cavidad se concluye que el mensaje fué reenviado. Con esta información, este elemento se da cuenta que el localizador débil de la referencia es inexacto. Por esta razón, la cavidad fabrica un mensaje de excepción de tipo *localizador-débil-inexacto*. Adicionalmente, la cavidad empaqueta en el mensaje un apuntador a si misma (i.e. un localizador exacto). Este mensaje es enviado diréctamente al objeto o a la aplicación cliente. Con la información de este mensaje el representante fuente actualiza su localizador débil para volverlo exacto. Después, el representante vuelve a enviar el mensaje de la invocación, y esto nos lleva al caso c, donde el localizador débil es exacto y la cadena es indirecta.

4 Comparación entre la técnica original y el esquema propuesto.

La diferencia básica entre las dos técnicas es la política respecto al recorte de las cadenas SSP. La técnica original busca minimizar el número de mensajes transmitidos por la red. Por esta causa, solamente aprovechan la invocación de métodos para realizar el recorte de cadenas SSP (§3.5). De esta manera, si no se producen invocaciones se pueden tener referencias arbitrariamente largas.
De otro lado, nuestra técnica aprovecha la mayoría de las situaciones para recortar las cadenas SSP. Estas situaciones son : El paso de un sitio a otro de las referencia que apuntan OMs, la migración de OMs, las invocaciones a métodos remotos y las invocaciones nulas que se hacen periódicamente. De esta manera, por lo general, las cadenas SSP son directas. En el peor de los casos, y solo momentáneamente, las cadenas pueden estar compuestas por tres parejas representante cavidad.

- **Ventajas de nuestro esquema sobre la técnica original**
  - Las invocaciones remotas se ejecutan más eficientemente sobre referencias directas que sobre referencias arbitrariamente largas.
  - La disponibilidad de los objetos es mayor utilizando referencias directas que referencias arbitrariamente largas.
- **Desventajas de nuestro esquema sobre la técnica original**
  - Nuestro esquema es más costoso que la técnica original debido a que continuamente se producen mensajes de control para recortar las cadenas SSP. Sin embargo, se puede seguir considerando una técnica escalable porque los mensajes de control son pequeños y los protocolos son relativamente sencillos.

5 Cómo se programarían objetos móviles : Ejemplo

Actualmente, uno de los problemas serios del WWW ocurre con la documentación HTML. Este problema consiste en que cuando un hipertexto HTML se mueve físicamente a otra máquina, todas las referencias (i.e. enlaces) que lo apuntaban quedan en estado inconsistente. Sin embargo, si estos documentos se modelaran como OMs el problema sería superado. Adicionalmente, si un buen número de los recursos del WWW (e.g. documentos HTML, applets, mensajes electrónicos, etc.) se modelaran como OMs, obtendriamos un sistema más homogéneo.
Lo que aquí pretendemos es dar una idea que puede ser interesante para el WWW; no pretendemos dar un modelo exacto. Por este motivo, las ilustraciones que siguen son un bosquejo.

En la figura 5, mostramos en código Java la forma como se definiría una clase de recursos del WWW representados como OMs. Estos objetos de aplicación extienden la funcionalidad de la clase *MobileObject*.

[4] En el proyecto [SDP92], la cavidad y el objeto destino pueden estar en sitios diferentes.

```
public class W3ObjectImpl extends MobileObject implements W3Object {

    String name;  // El nombre del objeto
                  // Ejemplo: "//isis.uniandes.edu.co/ri-de-home_ohtml"
    String owner; // El propietario del objeto
                  // Ejemplo: "ri-de@uniandes.edu.co"
    String group; // El grupo del objeto.
                  // Ejemplo: "estudiantes-informatica@uniandes.edu.co"
    Boolean mode[][];  // Los permisos de los usuarios sobre el objeto
                //          Read    Write    eXecution    Mobility
                //  owner     x       x                      x
                //  group     x                              x
                //  others    x

    byte content[]; // Contenido del objeto

    ...  // Los metodos los escribe el dise~nador del objeto como si el objeto
    ...  // fuera no migrante.
}
```

Figura 5: Recurso del WWW implementado como un objeto móvil.

Esta clase es proveida por nuestro sistema. En esta figura, el atributo *name* se refiere al nombre del OM. Este nombre es utilizado por entidades cliente para obtener una referencia a este recurso del WWW, independientemente de donde esté localizado. Es decir, el nombre es permanente y se refiere a su "host" de nacimiento y no a su máquina actual de residencia.
Un recurso también tiene un dueño y un grupo. Los permisos le sirven al dueño para controlar el acceso a su objeto. Adicionalmente a los permisos tradicionales tenemos el permiso de movilidad. En el ejemplo, el dueño y el grupo son los únicos que pueden mover el OM.

En la figura 6, aparece en código Java, la forma como un hipertexto HTML puede ser representado como un OM. Nótese que este objeto hereda de la clase *W3ObjectImpl*, que representa un recurso migrante del

```
public class HTMLDocument extends W3ObjectImpl {

    URL links_to_other_docs[];   // Enlaces a otros documentos

    int num_links_to;            // Número de enlaces a otros documentos

    URL links_from_other_docs[]; // Documentos que referencian este hipertexto

    int num_links_from;   // Número de documentos que referencian este hipertexto

    ...  // Los metodos los escribe el diseñador del objeto, como si este objeto
    ...  // fuera no migrante.
}
```

Figura 6: Documento HTML implementado como un objeto móvil.

WWW. Adicionalmente, la figura muestra como el documento se puede enriquecer teniendo información que pueda ayudar a su administración. En el ejemplo, se tienen los documentos que referencian al objeto, y además, los documentos referenciados por el objeto.

En la figura 7 aparece un bosquejo del código de lo que sería un navegador del WWW que accede objetos tipo documento HTML. Este ejemplo también ilustra el uso de OMs desde una aplicación cliente. En la figura 7 mostramos dos operaciones sobre un documento HTML. La primera corresponde a cargar el documento en

```
int event;
HTMLDocument current_doc;
String name_current_doc, target_machine, content;
...
while ( event != EXIT ) {
    if ( event == LOAD_DOCUMENT ) {
        // Ejemplo: name_current_doc == "/isis.uniandes.edu.co/ri-de-home_ohtml"
        current_doc = MO_Naming.lookup(name_current_doc);
        content = current_doc.getContent();
        pintar 'content' en la ventana del navegador.
      }
      else if ( event == MOVE_DOCUMENT ) {
        // Ejemplo: target_machine ==  "zeus.uniandes.edu.co"
        current_doc.move(target_machine);
      }
      else if (...) {
        ...
      }
}
```

Figura 7: Implementación de un navegador del WWW para objetos móviles tipo Documento HTML.

la ventana del navegador. Para esta operación se obtiene el contenido del documento invocando remotamente un método sobre el objeto[5]. Luego este contenido se despliega en la ventana del navegador.
La segunda operación, le permite a un usuario de un hipertexto (e.g. a su dueño) moverlo a otra máquina del WWW. Esta migración es transparente para otros clientes que estén accediendo el documento.

6 Trabajo relacionado

Debido a que Java RMI [Sun97] es una plataforma de vanguardia en cuanto a objetos remotos, vamos a analizar cuales son sus capacidades y límites en este campo. El mecanismo que utiliza RMI es básicamente el de RPC (i.e. *Remote Procedure Call*). En RMI se pueden pasar referencias a objetos RMI usando los argumentos o el valor de retorno de las invocaciones remotas. Posteriormente, el receptor de estas referencias las puede usar para hacer invocaciones remotas. Sin embargo, los objetos RMI no tienen capacidad de movimiento. Adicionalmente su servicio de nombres no soporta OMs.
Es pertinente aclarar que los "applets" de Java son tecnología de código móvil. Cuando una instancia de un "applet" se solicita en un sitio de la red, el código se mueve de un único sitio origen al sitio destino. En este último inicia su ejecución, siempre desde el mismo estado inicial. Además, desde este sitio el objeto no puede moverse más. Lo cual difiere del concepto de objeto móvil, donde un objeto conserva el estado entre migraciones y puede moverse a varios sitios.

El proyecto JavaParty [PZ97] propone una técnica para implementar OMs. Esta técnica es totalmente diferente a la técnica de cadenas SSP. Cuando un OM recibe la petición de migrar a un sitio, este objeto es sustituido por un representante. Si un representante recibe un llamado a un método, él produce una excepción que es recibida por el invocador en el retorno de la función. La nueva localización del OM es parte de la información de la excepción retornada. Con esta localización el invocador actualiza la referencia al OM y reintenta el llamado remoto.
Una desventaja de esta técnica tiene que ver con los objetos de alta movilidad. Supongamos que un OM se mueve n veces. También supongamos que una referencia no ha sido usada desde antes que el objeto decidiera moverse. Si se hace una invocación remota usando esta referencia, se deben hacer n reintentos antes de que el método sea ejecutado.

[5]Este objeto puede estar distante al sitio donde ejecuta el navegador.

7 Conclusiones

Hemos presentado un conjunto de mejoras sustanciales al esquema Cadenas SSP [SDP92]. La técnica original permite referenciar e invocar OMs en un sistema distribuido de área amplia (e.g. el WWW). Sin embargo, la forma en que se representa una referencia reduce la disponibilidad del objeto. Y además, conduce a complejos procedimientos de recuperación ante fallas y recolección de basura distribuida.
Lo que nosotros proponemos es cambiar las estrategias y protocolos para que se recorten las cadenas SSP cada vez que sea posible. Así, lo que se logra es maximizar la disponibilidad del objeto, ya que en el caso promedio una referencia depende de dos sitios: el sitio donde está el objetos y el sitio donde está el dueño de la referencia al objeto. En el peor caso, una referencia dependería de cuatro sitios. Esta política también facilitaría la recolección de basura.
Respecto a tolerancia a fallas, la técnica de cadenas SSP propone una búsqueda global, estructurando el universo en pequeños grupos de espacios terminantes. Sin embargo, una búsqueda exhaustiva no es un procedimiento escalable. Y por lo tanto, es muy cuestionable para un sistema distribuido de área amplia. Para remediar esta situación, proponemos que cada vez que sea necesario, el objeto le informe su localización al servidor de nombres que se encuentra en el sitio donde él nació. De esta manera, si después de una caida un cliente necesita comunicarse con un objeto, la solución es solicitarle al servidor de nombres respectivo, una referencia al recurso deseado.

Referencias

[BP96] A. Baggio y I. Piumarta. *Mobile Hosts Tracking and Resource Discovery.* Seventh ACM SIGOPS European Workshop "Systems Support for Worldwide Applications". Septiembre 1996.

[BAB96] S. Ben Hassen, I. Athanasiu y H.E. Bal. *A Flexible Operation Execution Model for Shared Distributed Objects.* Proc. Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'96), pp. 30-50, California, USA. ACM. Octubre 1996.

[BC96] K.A. Bharat y L. Cardelli. *Migratory Applications.* Proc. Eighth Annual Symposium on User Interface Software and Technology, Pennsylvania, USA. ACM. Reporte de Investigación No. 138, SRC of Digital Equipment Corporation. Noviembre 1995.

[MGLS94] M. Makpangou, Y. Gourhant, J.P. Le Narzul y M. Shapiro. *Fragmented Objects for Distributed Abstractions.* En T.L. Casavant y M. Singhal (eds). Readings in Distributed Computing Systems, pp. 170-186. IEEE Computer Society Press. 1994.

[OMG95] Object Management Group. *The Common Object Request Broker: Architecture and Specification, Version 2.0.* Reporte Técnico. Julio 1995.

[PZ97] M. Philippsen y M. Zenger. *JavaParty - Transparent Remote Objects in Java.* PPOPP Workshop on Java for Science and Engineering Computation, Nevada, USA. ACM. Junio 1997.

[SDP92] M. Shapiro, P. Dickman y D. Plainfossé. *SSP Chains : Robust, Distributed References Supporting Aciclic Garbage Collection.* Reporte de Investigación No. 1799, INRIA. Noviembre 1992.

[SPFA94] M. Shapiro, D. Plainfossé, P. Ferreira y L. Amsaleg. *Some Key Issues in the Design of Distributed Garbage Collection and References.* Proc. Unifying Theory and Practice in Distributed Systems, Dagstuhl, Alemania. Septiembre 1994.

[SHT96] M. van Steen, F.J. Hauck y A.S. Tanenbaum. *A Model for Worldwide Tracking of Distributed Objects.* Proc. TINA'96 Conference, Heidelberg, Alemania. Septiembre 1996.

[Sun97] Sun Microsystems. *Java Remote Method Invocation Specification, Revision 1.4.* Febrero 1997.