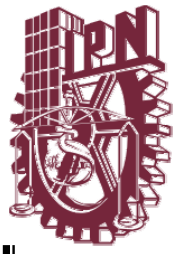


INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Sistema Multi-Agente para monitorizar el control de concurrencia de Transacciones Anidadas sobre grupos de dispositivos móviles

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
Jorge Martínez Muñoz

DIRECTOR:
Dr. J. Matías Alvarado M.

México, D.F. Octubre de 2007.

SIP-14

INSTITUTO POLITECNICO NACIONAL

SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 18:00 horas del día 14 del mes de Diciembre de 2005 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"SISTEMA MULTI-AGENTE PARA MONITORIZAR EL CONTROL DE CONCURRENCIA DE TRANSACCIONES ANIDADAS SOBRE GRUPOS DE DISPOSITIVOS MÓVILES"

MARTÍNEZ	**MUÑOZ**	**JORGE**
Apellido paterno	materno	nombre(s)

Con registro:

B	0	1	1	3	8	6

aspirante al grado de: ***MAESTRO EN CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. Adolfo Guzmán Arenas

Secretario

Dr. Hugo César Coyote Estrada

Primer vocal
(Director de Tesis)

Dr. Matías Alvarado Mentado

Segundo vocal

Dr. Felipe Rolando Menchaca García

Tercer vocal

M. en C. Sergio Sandoval Reyes

Suplente

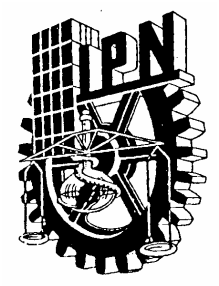
M. en C. Rubén Peredo Valderrama

INSTITUTO POLITECNICO NACIONAL SECRETARIA DE INVESTIGACION Y POSGRADO 20 ENE 2006 RECIBIDO DIRECCION DE POSGRADO

EL PRESIDENTE DEL COLEGIO

Dr. Hugo César Coyote Estrada

INSTITUTO POLITECNICO NACIONAL CENTRO DE INVESTIGACION EN COMPUTACION DIRECCION



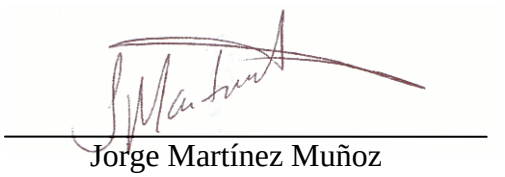
INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESION DE DERECHOS

En la Ciudad de México D.F., el día 4 del mes de Octubre del año 2007, el que suscribe Jorge Martínez Muñoz, alumno del Programa de Maestría en Ciencias de la Computación con número de registro B011386, adscrito al Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. J. Matías Alvarado Mentado y cede los derechos del trabajo intitulado "Sistema Multi-agente para monitorizar el control de concurrencia de Transacciones Anidadas sobre grupos de dispositivos móviles", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección: jorge_mtz_m@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Jorge Martínez Muñoz

RESUMEN

El paradigma y la tecnología de Sistemas Multi-Agente, actualmente, es una de las contribuciones más representativa de la Inteligencia Artificial Distribuida. Los sistemas multi-agente se concibieron bajo la posibilidad de dividir problemas altamente complejos, en fragmentos que son procesados por entidades autónomas con comportamiento colectivo.

Por otro lado, los avances en la tecnología de comunicaciones inalámbrica, han dado pie a un nuevo estilo para hacer computación. Actualmente, los usuarios y consumidores disponen de una gama de dispositivos y servicios donde la palabra clave es *movilidad*. Así, los elementos intrínsecos a la operación de los dispositivos y a los canales de comunicación, requieren de esquemas flexibles para el manejo de datos. El procesamiento de transacciones, en particular, se orienta a la preservación de la consistencia e integridad de los datos.

En la presente tesis se modela e implementa un monitor de procesamiento de Transacciones Anidadas sobre dispositivos móviles. Los objetivos son:

- Formalizar el modelo de transacciones anidadas utilizando la Lógica de Interacción.
- Implantar dicho modelo, en un monitor, mediante un sistema multi-agente.
- Experimentar con el monitor, con la expectativa de asegurar la consistencia e integridad de los datos en un entorno tecnológico de computación móvil, operando sobre redes inalámbricas.

ABSTRACT

Multi-agent systems, as a paradigm and technology, are currently one of the most representative contributions of Distributed Artificial Intelligence. Multi-agent systems were conceived under the possibility of dividing complex problems into work-pieces, which are processed by autonomous entities.

On the other hand, advances in wireless communications, pushed the rise of a new style in computing technology. Nowadays, users and consumers make use of a complete set of devices and services where *mobility* is the key point. There, intrinsic elements to device operation and communication channels require flexible mechanisms for data handling. Transaction processing, in particular, is oriented towards preservation of data consistency and integrity.

In this thesis, it is modeled and implemented a nested transaction processing monitor over mobile devices. The objectives are:

- To formalize the nested transaction model using Logic of Interaction.
- To implement the above model in a monitor using a multi-agent system.
- To experiment with the monitor, with the expectative of ensuring data consistency and integrity in a mobile computing environment, operating over a wireless network.

CONTENIDO

Lista de Tablas

Lista de Figuras

1 Introducción

A continuación se describe brevemente el contexto del sistema que se desarrolla en la presente tesis. Asimismo, se enuncia el problema a tratar, los objetivos y el alcance del trabajo.

Los Sistemas Multi-Agente (SMA) son la contribución más representativa de la Inteligencia Artificial Distribuida. Su origen es ocasionalmente asociado al comportamiento que exhiben diferentes comunidades de animales en el desarrollo de sus respectivas tareas, e incluso se ha establecido correspondencia con las actividades sociales del ser humano. El enfoque Multi-Agente se concibió bajo la posibilidad de dividir problemas altamente complejos y de naturaleza distribuida. Cada fragmento es tratado por un agente autónomo que interactúa y se comunica con el resto del grupo durante el proceso de solución.

Se habla también de una nueva etapa en la programación de sistemas de información. Una etapa donde los alcances de la Programación Orientada a Objetos, son superados con las habilidades de los Agentes inteligentes. Si bien, en el ámbito comercial no se percibe el potencial completo de esta tecnología, ya se cuenta con plataformas lo suficientemente maduras para el despliegue de soluciones basadas en SMA. La fortaleza de estos sistemas se relaciona también con la teoría que fundamenta su operación.

Una de las áreas de interés para la implementación de SMA, se localiza en el campo del Cómputo Móvil. La disponibilidad tecnológica de terminales portátiles que se comunican por vía inalámbrica, dio pie al surgimiento de nuevos estilos de hacer computación. Sin embargo, elementos intrínsecos a la operación de los dispositivos y a los canales de computación, requieren de esquemas flexibles para el manejo de datos. El procesamiento de transacciones en particular, se orienta a la preservación de la consistencia e integridad de los datos. Los escenarios más complejos (y por ende, más apegados a la realidad) son aquellos donde la información es accedida de forma concurrente por múltiples clientes. Adicionalmente, una solución adecuada debe tomar en cuenta que los datos se encuentran alojados en distintas ubicaciones.

En este punto se revela una intersección de los tres elementos, Sistemas Multi-Agente, Cómputo Móvil y Procesamiento de Transacciones, que es el motivo del trabajo desarrollado en la presente tesis. La problemática por atender es: **coordinar un grupo de agentes autónomos, para monitorizar el procesamiento de transacciones que acceden a datos residentes en dispositivos móviles.**

Desde la perspectiva del significado profesional que representa este trabajo, se enumeran los siguientes **objetivos**:

a) aterrizar un esquema formal (Lógica de Interacción) en una implementación práctica de sistemas multi-agente,
b) experimentar con un modelo de transacciones (Anidadas) que en la teoría, ha demostrado mejoras en el procesamiento distribuido de datos con respecto a otros modelos más tradicionales y
c) aplicar los dos conceptos anteriores en un ambiente en auge (Cómputo Móvil) cuyas implicaciones tecnológicas demandan esquemas flexibles en el tratamiento de la información.

La revisión del marco teórico y tecnológico (**Cap. 2**) es necesaria para contextualizar este trabajo con respecto a desarrollos anteriores o en curso; así también, para clarificar los conceptos básicos sobre los cuales se basa la arquitectura de los agentes que integran al monitor implementado (**Cap. 3**). En su definición más general, se espera que un monitor de transacciones atienda escenarios donde existen cientos o miles de clientes solicitando acceso a los datos. En contraste, en esta tesis se estudia el comportamiento y posibilidades de grupos de trabajo reducidos. El análisis de resultados (**Cap. 4**) se realiza a partir de pruebas con cinco a siete clientes concurrentes, desde dispositivos móviles reales y emulados. Finalmente, se presenta una discusión y la perspectiva del trabajo (**Cap. 5**) a partir de los resultados alcanzados.

2 Marco Teórico y Tecnológico

En este capítulo se presentan definiciones y discusión sobre las bases teóricas del sistema implementado. Estas son: Cómputo Móvil, Teoría de Transacciones, Sistemas Multi-Agente y la Lógica de Interacción; para esta última se introducen detalles sobre su implementación en la plataforma de agentes.

2.1 Cómputo Móvil

Los avances en redes de comunicación inalámbrica y aplicaciones portátiles de información, han introducido un nuevo paradigma conocido como cómputo móvil. Los usuarios ya no están limitados a ubicaciones fijas; con la ayuda de dispositivos móviles, como teléfonos celulares, asistentes digitales personales y computadoras portátiles, es posible acceder a la información sin importar la ubicación geográfica. La Figura 1 ilustra un ambiente de cómputo móvil; éste se compone de:

1. *Hosts* Fijos. Se trata de servidores de recursos que se localizan en ubicaciones establecidas; esto se debe principalmente a la cantidad y al tamaño de los procesos que se ejecutan concurrentemente. Un subconjunto de los *Hosts* fijos se denomina Estaciones de Soporte Móviles (ESS) y tienen la responsabilidad de proporcionar el servicio de interfaz inalámbrica a los *Hosts* Móviles. Su rango de alcance (limitado a cierto radio geográfico) se conoce como Celda. Tanto los *Hosts* fijos como las Estaciones de Soporte Móvil están conectadas por medio de redes de comunicaciones públicas o privadas.

2. *Hosts* Móviles. Este grupo comprende cualquier dispositivo capaz de establecer un enlace inalámbrico (o celular) a una Estación de Soporte Móvil. De esta forma, el dispositivo se enlaza a la red fija. Las ESS son responsables de todos los *hosts* que se encuentran dentro de su rango de alcance; sin embargo, cuando un *host* móvil abandona la celda, otra estación debe proporcionar el servicio.

Un *Usuario Móvil* es aquel quien se conecta a la red de comunicaciones fija por medio de su *Host* móvil. Desde la perspectiva del usuario, sería altamente deseable mantener la conexión activa mientras aquél se desplaza. Los problemas con ambientes móviles se originan precisamente cuando el dispositivo pierde la conexión de datos por cualquier razón. Las más comunes son: baterías bajas, ubicación actual del dispositivo (por ejemplo, el usuario entra a una estación del Metro) e inestabilidad de los enlaces.

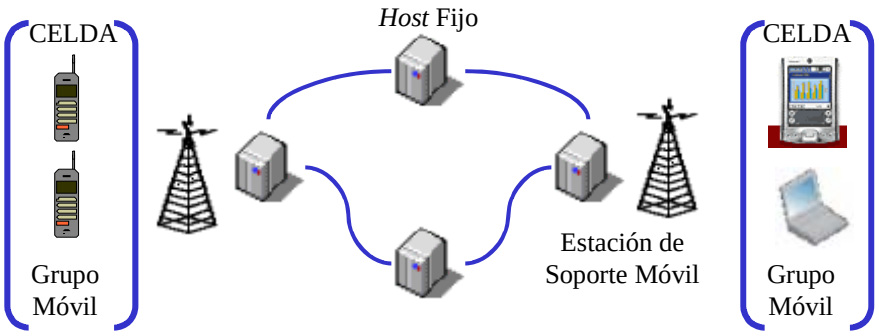


Figura 1. Ambiente de Cómputo Móvil con dos Celdas de servicio

La implementación de aplicaciones para cómputo móvil, presenta retos ante la solución de un problema distribuido por naturaleza y que está ubicado en ambientes altamente dinámicos [Zaslavsky98].

2.1.1 Tecnología inalámbrica

La comunicación entre *hosts* fijos y móviles, así como entre estos últimos, son el resultado de la evolución científica-tecnológica desde aquellos experimentos conducidos por Nikola Tesla en 1893 y Guglielmo Marconi en 1896 con los primeros telégrafos inalámbricos [Engelmann00]. Los teléfonos celulares pueden ser considerados la versión contemporánea de dichos telégrafos. Las compañías de hoy pueden contar con oficinas comunicadas totalmente de forma inalámbrica por medio de sistemas basados en el estándar IEEE 802.11. Desde la perspectiva del *Usuario Móvil*, las tecnologías más representativas [Stallings01] para enlazarse de forma inalámbrica, se mencionan a continuación.

- La comunicación por *Infrarrojos*, que se lleva a cabo con transmisores y receptores de luz infrarroja modulada. Se diferencia de la transmisión por microondas por las limitantes en alcance (hasta 20 m. en las mejores condiciones) y porque no es posible traspasar superficies sólidas; de hecho, los transmisores y receptores deben estar alineados en su línea de vista. Por otro lado, no existe el problema legal para asignación de frecuencias, ya que no se requiere obtener licencia alguna. Una computadora portátil puede emplear un teléfono celular como módem si ambos están habilitados con puertos de comunicación infrarroja.

- La esencia de la comunicación *Satelital* reside en antenas ubicadas en órbitas estables alrededor del planeta. Estaciones en tierra se enlazan por medio de una o varias de estas antenas. Se identifican tres tipos de servicios: Fijos, de Difusión y Móviles. El costo actual y los requerimientos en equipo

especializado, restringen el uso de esta tecnología a: difusión de medios masivos, aplicaciones militares e institucionales (gubernamentales), experimentación, y solo bajo condiciones particulares el uso de parte de aficionados. Un caso particular lo representan los Sistemas de Posicionamiento Global (GPS) que actualmente se encuentran disponibles como productos para el consumidor final.

- El *Sistema Global para Comunicaciones Móviles (GSM)* nació como una iniciativa europea para integración de las *Redes Celulares Inalámbricas*. El principio de operación se encuentra en las comunicaciones satelitales; no obstante, el diseño de la arquitectura GSM identifica además: Subsistemas de Estaciones Base que ofrecen cobertura a su respectiva Celda; Subsistemas de Red para enlazar redes celulares con redes de telecomunicaciones públicas y la Estación Móvil del usuario, misma que cuenta con un dispositivo portátil, el Módulo de Identidad del Subscriptor (SIM) –que almacena información específica para el uso del suscriptor y para autorización de acceso a la red.

- Las especificaciones del grupo IEEE 802.11 (relevante trabajo de liderazgo) se adoptaron como el estándar en la mayoría de las *Redes de Área Local Inalámbricas* (W-LAN) que se emplean actualmente. La creciente popularización de este tipo de redes está ligada de forma estrecha con el diseño inteligente de nuevas instalaciones, así como con las condiciones especiales en edificaciones donde el tendido de cables no es práctico. De forma análoga a los concentradores (*hubs*) LAN alambrados, existe equipo con interfaces inalámbricas que cubren celdas y ofrecen servicios como: extensión de la red local, interconexión de edificios y redes de trabajo par-a-par (sin un servidor central).

- El *Protocolo Inalámbrico de Aplicación (WAP)* es un estándar desarrollado por el *WAP Forum* para proporcionar a terminales inalámbricas, acceso a servicios de información, incluyendo Internet. Se distingue notablemente por tomar en consideración las capacidades limitadas de los dispositivos en lo que respecta a presentación, almacenamiento y procesamiento. Asimismo, el modelo de programación está basado en el modelo de la *World Wide Web* y opera con un lenguaje de marcas (*tags*) adherido a XML.

- La idea de contar con enlaces siempre activos, de corto alcance y con mayor universalidad en su aplicación, llevó al desarrollo de *Bluetooth*. Se trata de un microchip que proporciona capacidades inalámbricas en una frecuencia de radio abierta. Con un alcance de 10 metros y un ancho de banda de 720 kilobits por segundo, *Bluetooth* está orientado a soportar aplicaciones que van desde la transmisión de datos hasta la difusión multimedia, e incluyen

el control e interacción entre dispositivos de cómputo y comunicaciones con aparatos electrodomésticos. Los servicios se pueden categorizar en puntos de acceso para voz y datos, reemplazo de cables y redes *ad-hoc*.

2.1.2 ¿Cómputo móvil, ubicuo o penetrante?

La literatura al respecto muestra diferentes puntos de vista en los que, para estos tres conceptos se presentan tanto definiciones bien diferenciadas como intersecciones entre ellos. Por ejemplo, en [Satyanarayanan93] se menciona que "la proliferación de computadoras portátiles aunada a la tecnología inalámbrica, proveerá la base **penetrante** para cómputo **móvil** ... la telecomunicación será verdaderamente **ubicua**."

Otra definición para cómputo móvil [Forman94] se va directamente al uso de una computadora portátil habilitada con acceso inalámbrico a una red de comunicaciones. En contraste, en [Racherla96] se considera que la raíz del **cómputo móvil** está precisamente en la movilidad de los usuarios. Allí, en un escenario ideal el cómputo se realizaría de forma transparente sin importar la ubicación del usuario, su desplazamiento, el ambiente y las redes de comunicación disponibles. Asimismo, se establece que el **cómputo móvil** no es esencialmente inalámbrico; la razón: el usuario no siempre está en movimiento

Un punto común en la literatura sobre **cómputo móvil** se encuentra en la *adaptabilidad* como elemento clave. El objetivo es tratar con distintas y variantes capacidades de cómputo, esquemas de comunicación, así como el contexto físico y virtual del usuario. Los sistemas deben adaptar la distribución del trabajo delegando funciones en compuertas *middleware* [Mazer95]. En este mismo trabajo se presenta la filosofía del **cómputo ubicuo**, que apunta a enriquecer la experiencia computacional incrustando dispositivos de diferentes tipos en el ambiente para proporcionar acceso más personalizado a la información.

En 1991, Mark Weiser escribió a manera de predicción, que "las tecnologías más profundas son aquellas que desaparecen, se difuminan en la vida diaria en tal medida que se vuelven indistinguibles de ella" [Weiser91]. A este artículo se le considera la formalización de lo que ahora se conoce como **cómputo ubicuo**. El mismo autor en [Weiser93] explica que el acceso a la computadora se realizará en cualquier lado, desde la palma de la mano hasta las paredes. La meta es obtener disponibilidad no *intrusiva* de computadoras a través del ambiente físico y además, invisible al usuario. El despliegue y acceso a información serán de tal forma que ni siquiera se requerirá un asistente digital personal (PDA) porque la información estará donde sea.

En [Borriello02] se presenta el **cómputo ubicuo** en forma de ejemplos. En el cómputo tradicional, un usuario debe trasladarse entre dos ubicaciones. Para llegar a tiempo, requiere consultar algún servicio de monitorización del tráfico. Esto implica la introducción manual de datos y el cálculo de tiempo con los resultados obtenidos. En un ambiente de cómputo ubicuo, se tiene un agente autónomo encargado de realizar las consultas; si así se requiere, el agente puede ajustar automáticamente la agenda electrónica del usuario y enviar las alarmas necesarias. Observamos que el agente en cuestión puede ser de hardware o de software, actualmente.

De acuerdo a [Park04], el **cómputo ubicuo** se caracteriza por invisibilidad de los objetos, consciencia del contexto y adaptación proactiva, movilidad y tecnologías de comunicación. En un ambiente ubicuo la información contextual del entorno se adquiere de sensores, GPSs y de la misma gente. En [Kindberg02] se hace hincapié en dos características clave del cómputo ubicuo:

1) la integración física entre nodos de cómputo y el mundo físico,
2) la operación espontánea de los componentes en un ambiente dinámico, esto es, interacción constante a pesar de los cambios en identidad y funcionalidad de tales componentes.

A estas puntualizaciones debe añadirse el que:

3) no es intrusivo y es invisible [Weiser93].

El concepto de **cómputo penetrante** es expresado en [Freeland01] como una visión del futuro de la información digital. En ella, tanto aplicaciones como servicios estarán altamente personalizados en función del usuario. El acceso será por medio de dispositivos y redes de comunicación integrados. Algunos ejemplos de los servicios: banca móvil, asistencia en el camino (mapas y directorios), noticias, electrodomésticos inteligentes, entre otros.

En [Bacon02] se presenta al **cómputo penetrante** como la evolución natural de los sistemas distribuidos; no obstante, se hace referencia al trabajo de Weiser como el inaugurador del concepto. En particular, se menciona que la penetrabilidad estará beneficiada por ambientes plenos en sensores tales como los edificios inteligentes. También se deja entrever la orientación a servicios para los consumidores.

Finalmente, en [Niemela04] se define al **cómputo penetrante** como la incrustación de tecnología computacional en el ambiente de todos los días. Establece la diferenciación con respecto al **cómputo ubicuo** en el sentido que, este último está más orientado a poblar el ambiente con dispositivos. Por ello puede verse como prerrequisito para establecer penetrabilidad de servicios y aplicaciones, donde debe existir además, aprovechamiento inteligente de los recursos a favor del usuario. Un ambiente penetrante se identifica por: consciencia del contexto, movilidad e integración.

La revisión de conceptos lleva a remarcar los siguientes elementos:

- Tanto en cómputo móvil como ubicuo, se hace hincapié explícito en la adaptabilidad como elemento básico de los sistemas de información.
- El cómputo ubicuo está asociado con la noción de invisibilidad en contraste al móvil y el penetrante.
- El cómputo ubicuo se basa mayormente en la disponibilidad de comunicaciones inalámbricas; mientras que el móvil y el penetrante integran asimismo la infraestructura alambrada actualmente disponible.
- El cómputo móvil puede considerarse elemento de soporte para el despliegue y realización de los otros dos.
- El cómputo ubicuo se percibe a su vez como el facilitador para lograr penetrabilidad; existen además elementos compartidos como la consciencia del contexto y la movilidad.
- El cómputo penetrante es el único de los tres que se explicita a sí mismo no sólo como una tecnología, sino también como un negocio.

A la luz de las de los conceptos revisados, es posible ubicar la presente tesis en el contexto del Cómputo Móvil; principalmente, porque el sistema está orientado a la administración de piezas de trabajo distribuidas y a la adaptabilidad de entidades autónomas en las decisiones necesarias durante el procesamiento de transacciones. Asimismo, no se restringe la operación a dispositivos que se comunican por vía inalámbrica. Existe, por otro lado, transparencia en la ubicación de los agentes que participan en una transacción.

2.2 Teoría de Transacciones

El concepto de *Transacción* puede ser encontrado en casi cualquier tipo de procesos de negocios actualmente modelado por técnicas orientadas a objetos. En [Orfali95] se le da especial significado al Procesamiento de Transacciones como un hito dentro de la evolución de la tecnología Cliente/Servidor.

Una definición de [Coulouris02] dice que una transacción opera como un mecanismo para preservación de la consistencia en un conjunto de objetos de trabajo. Una creencia común acerca del origen de las transacciones, establece que provienen de la teoría de Sistemas de Administración de Bases de Datos. Su nacimiento se dio por la necesidad de los usuarios por ejecutar, como unidad, un conjunto de operaciones sobre una base de datos. La idea de la transacción, sin embargo, se introdujo en la teoría de Sistemas Distribuidos bajo la forma de Servidores Transaccionales de Archivos [Mitchell82]. Aquí el objetivo consistía en proporcionar a usuarios concurrentes, servicios de acceso a archivos sobre una red de comunicaciones. El requisito era proteger la consistencia contra múltiples solicitudes que podían ser de lectura o escritura.

En la presente tesis, el objetivo no sólo es controlar el acceso a datos, sino también coordinar un grupo de dispositivos móviles que incluyen tanto unidades lógicas de información como dispositivos físicos que acceden a ésta última.

2.2.1 Propiedades *ACID*

Los modelos de transacciones se encuentran estrechamente relacionados con cuatro propiedades básicas:

- *Atomicidad*, es decir "todo o nada" e implica que el conjunto de operaciones que componen una transacción debe ser ejecutado como un todo. Esta propiedad adquiere mayor relevancia en las transacciones distribuidas, donde por medio de un protocolo de compromiso atómico, un coordinador interactúa con los participantes de la transacción para asegurar la ejecución o cancelación global.
- Aunque la *Consistencia* está considerada como una propiedad, en realidad hace referencia al estado de la base de datos; sin embargo, es responsabilidad de la transacción tomar la base de datos en un estado consistente y dejarla en un estado similar.
- El *Aislamiento* (*Isolation*) significa que una transacción debe verse a sí misma como la única en ejecución en un momento dado. Junto a la *Consistencia*, ambas propiedades son representativas de las tareas que lleva a cabo el Mecanismo de Control de Concurrencia y la coordinación de los accesos a datos (generalmente múltiples y simultáneos) por parte de los clientes.
- La *Durabilidad* representa la persistencia de datos; esto es, una vez que la transacción ha sido confirmada, los cambios sobre la base de datos no pueden ser sino por medio de la ejecución de una segunda transacción. Es una responsabilidad dividida entre el Sistema Manejador de Bases de Datos y el Sistema Operativo anfitrión, ya que implica acceso al hardware para asegurar la efectiva actualización de los objetos modificados.

Proteger las propiedades *ACID* es la tarea definitiva a realizarse en el procesamiento de transacciones. Sin embargo, recientes trabajos [Kempster99], [Seifert03], [Gama03] relacionados con el control de concurrencia, proponen el relajamiento de las condiciones de aislamiento. El beneficio es un mejor tratamiento en un mayor número de clientes concurrentes. A ellos se les permite el acceso a los recursos compartidos evitando introducir conflictos adicionales en las operaciones de lectura/escritura.

2.2.2 Transacciones distribuidas

Una transacción plana puede ser definida como un conjunto de operaciones atómicas sobre un grupo de objetos en una base de datos. Estas operaciones están organizadas bajo un orden parcial [Gray93]. Se definen tres operaciones para demarcar el contexto de una transacción: *Begin, Commit, Abort.* Las dos últimas se ejecutan al final para indicar que la ejecución tuvo éxito o no.
Cada petición que pertenece a una transacción plana se ejecuta sobre datos que residen en el mismo servidor. En una Transacción Distribuida, existen dos o más servidores que proporcionan los recursos a consultar o modificar por la transacción; por ello, surge la necesidad de tener un Coordinador a cargo de administrar el resultado exitoso o fallido de todos los participantes. Esto se lleva a cabo por medio de un protocolo de compromiso atómico como puede ser el Compromiso a Dos Fases (*Two-Phase Commit Protocol* 2PC) [Özsu99].

2.2.3 Transacciones anidadas

La idea de Transacciones Anidadas extiende al concepto Distribuido al permitir que otras transacciones (conocidas como hijos) nazcan dentro del contexto de una transacción padre [Coulouris02][Moss85]. De esta forma, se obtiene un árbol de nodos que ejecutan operaciones de lectura y/o escritura al tiempo que son capaces de engendrar sub-transacciones internas. La única limitante para la profundidad del árbol son los recursos de cómputo disponibles. Sin embargo, se descarta la restricción de completar todas las operaciones en el mismo lugar. Cada nodo hijo puede ejecutarse en distintas ubicaciones, en otras palabras, las transacciones anidadas son distribuidas en esencia. En [Gama02][Gama04] se concluyó que las transacciones anidadas se caracterizan por condiciones especiales según las cuales no es posible cumplir con todas las propiedades ACID. Con excepción del nodo raíz, los demás nodos fallan al asegurar la *Durabilidad*. La razón es que cualquier cambio en la base de datos, realmente se lleva a cabo hasta que el nodo raíz decide confirmar la transacción. Un ejemplo de un árbol se ilustra en la Figura 2.

Ciertas características de las Transacciones Anidadas las vuelven adecuadas para aplicaciones móviles que involucran enlaces inalámbricos o celulares. Por ejemplo, una transacción compuesta con hijos anidados, ofrece mejor desempeño en concurrencia y tolerancia a fallos. Esto es, los hijos pueden ser ejecutados de forma paralela donde cada uno mantiene responsabilidad local para confirmar o abortar. No obstante, ninguna de estas operaciones se refleja sobre el contenido de la base de datos hasta que el nodo raíz confirma o aborta. Algunas otras características de las Transacciones Anidadas incluyen la opción de éxito como requisito de los

nodos hijo. De acuerdo con esto el nodo raíz es capaz de confirmar, aunque alguna de las ramas haya fallado durante la ejecución.

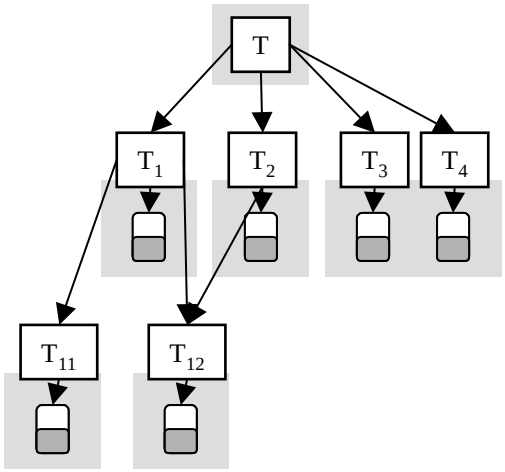


Figura 2. Transacción Anidada. El Nodo Raíz T engendra cuatro hijos. T_1 engendra dos hijos más resultando en un total de cuatro Nodos Hoja. Puede verse que T_{12} opera como sub-transacción tanto para T_1 como T_2. T_3 y T_4 operan físicamente en el mismo nodo

2.2.4 Transacciones anidadas móviles

Este concepto se introduce en [Gama04]. Se trata del resultado de combinar los conceptos de transacciones anidadas con un grupo de *hosts* móviles. Un dispositivo móvil representa cada nodo del árbol de la transacción. Una diferencia notoria entre el modelo general de Transacciones Anidadas y propuestas más recientes, se localiza en el acceso de lectura/escritura. En [Moss85] sólo los nodos hoja pueden acceder a los objetos de la base de datos mientras que la definición en [Gray93] habilita a los nodos intermedios para ejecutar también cualquiera de esas dos operaciones. El enfoque desarrollado en el presente trabajo se apega a la primera definición en el sentido que las operaciones se ejecutan sólo en el nivel más bajo (hoja); asimismo, se consideran las siguientes características: los nodos hijo tienen sólo un padre y cada nodo de la transacción puede estar ubicado en un *host* móvil o bien, puede compartir el dispositivo con otro nodo (ver Figura 3).

2.3 Control de Concurrencia

En principio, un cliente es responsable de solicitar la ejecución de operaciones sobre un servidor. La consecuencia natural de múltiples solicitudes, genera un problema de control de concurrencia sobre los objetos de interés. Del tipo de operaciones que componen la transacción, lectura o escritura, depende que el

servidor autorice acceso al recurso solicitado. A continuación se describen los esquemas empleados para tal fin.

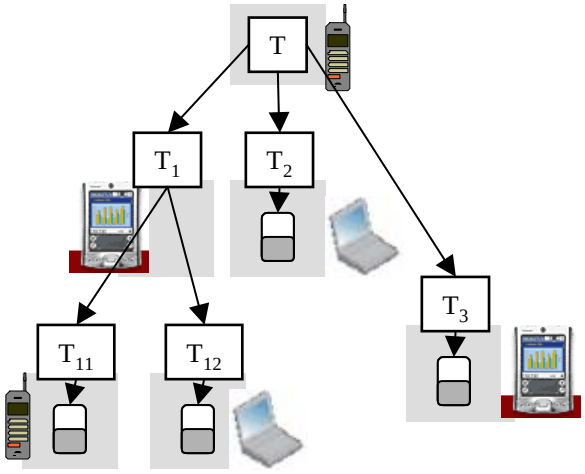


Figura 3. Transacción Anidada Móvil

2.3.1 Basado en candados

En este tipo de control, el servidor donde residen los datos establece candados o bloqueos para cada objeto que será accedido por las operaciones de la transacción. Dependiendo si la operación es de lectura o escritura, el recurso se bloquea para acceso compartido o exclusivo respectivamente. La presencia de candados previene que otras transacciones modifiquen los datos.

En el Bloqueo a Dos Fases *(Two Phase Locking 2PL*), una primera etapa permite a la transacción adquirir todos los candados necesarios sobre los objetos que desea leer o escribir. En la segunda fase se liberan todos los candados adquiridos. La implementación se lleva a cabo por medio de un *Administrador* que protege el acceso compartido cuando las transacciones ejecutan operaciones de lectura; en caso contrario, las transacciones deben retrasarse hasta la liberación de los candados para poder escribir sobre el objeto deseado.

2.3.2 Optimista

El uso de candados acarrea ciertas desventajas que impactan como sobrecarga de trabajo en el mantenimiento y administración de los mismos. También pueden presentarse ciclos de abrazo mortal donde existen transacciones esperando respectivamente la liberación del objeto por parte de otra transacción. A la propuesta alternativa de [Kung81] se le llama Control de Concurrencia Optimista y

se basa en la suposición que los conflictos por acceso simultáneo a un mismo objeto, son eventos de baja probabilidad. Esta observación se refuerza con modelos matemáticos donde se ha estimado que: para un sistema con N transacciones concurrentes, cada una solicita 20 candados simultáneamente, transcurre un tiempo T entre cada solicitud y la base de datos cuenta con 1 millón de objetos, la probabilidad de un abrazo mortal es apenas de 4 en 10,000 [Bernstein97].

El control Optimista permite que las transacciones procedan como si no existiera la posibilidad de conflicto con otras transacciones y, en caso de presentarse alguno, la transacción se aborta. El control distingue tres fases: a) trabajo, en la cual se emplean versiones tentativas de los datos; b) validación, donde se detectan los conflictos que hayan emergido por acceso concurrente y c) actualización, si la transacción es válida, se cambian permanentemente los datos.

2.3.3 Basado en estampas de tiempo

Este control etiqueta cada operación con la estampa de tiempo en la cual se ejecuta y la valida en el instante en que se lleva a cabo. Si no es posible confirmar la operación, toda la transacción se aborta. Asimismo, cada transacción T_i en el sistema, obtiene una estampa de tiempo única, denotada por $TS(T_i)$. Esta etiqueta es determinada por el sistema de base de datos antes de que la transacción T_i inicie su ejecución. Si se asigna un TS a la transacción T_i y una nueva transacción T_j se inicia en el sistema, entonces $TS(Ti) < TS(T_j)$. Las estampas de tiempo de las transacciones establecen un orden de serialización. De esta forma, si $TS(T_i) < TS(T_j)$, el sistema debe asegurar que el cronograma generado sea equivalente a un cronograma serial en donde T_i aparezca antes que T_j. Aunque se asegura la ausencia de abrazos mortales, una desventaja de este control es que puede desencadenar abortos en cascada de transacciones en conflicto.

2.4 Sistemas Multi-Agente

Los Sistemas Multi-Agente han emergido como un enfoque alternativo a la solución de problemas distribuidos. De hecho, este tipo de sistemas se derivan de los esfuerzos en investigación dentro del campo de la Inteligencia Artificial Distribuida. Fue precisamente el desarrollo de mecanismos de intercambio de mensajes entre sistemas computaciones, el cual impulsó la idea de modelar cada entidad como un agente.

Las características básicas en el comportamiento esperado de un agente son:

- Autonomía, la cual implica que el agente (hasta cierto punto) mantiene control sobre su estado interno y la forma en que actúa sobre su ambiente;

- Habilidad social, el agente debe ser capaz de interactuar y comunicarse con otras entidades durante el proceso de solución de problemas;
- Aprendizaje, esto es, el agente evoluciona a través de su ciclo de vida y adquiere nuevos conocimientos y habilidades.

Características adicionales del agente y los sistemas multi-agente incluyen aquellas relacionadas con: razonamiento, movilidad y persistencia. También se espera que los agentes ejecuten operaciones sensoriales sobre su ambiente, así como adaptabilidad para tratar con eventos inesperados. El diseño de los agentes debe permitirles alcanzar sus metas bajo condiciones descritas para su aprovisionamiento de información (conocimiento/creencias) [Wooldridge01].

Por los aspectos antes mencionados, se vuelve factible la implementación de sistemas multi-agente sobre grupos de dispositivos móviles, esto es, uno o más agentes por dispositivo que interactúan para completar un conjunto de metas (ver Figura 4). Esta es la base para un mecanismo tolerante a fallos que resuelva el problema de pérdidas de comunicación, muy común en dispositivos dentro de un ambiente móvil.

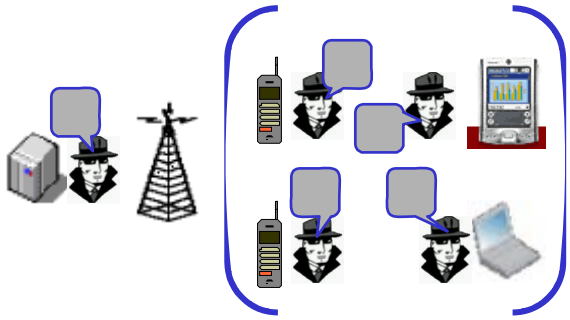


Figura 4. Sistema Multi-Agente implementado sobre un grupo de dispositivos móviles

2.4.1 Sistemas multi-agente en cómputo móvil y procesamiento de transacciones.

Los aspectos subyacentes de implementar Sistemas Multi-Agente en Cómputo Móvil se han tratado en [Finin02] y [Loke02], incluso éste último menciona un enfoque a bases de datos similar al que se desarrolla en el presente trabajo. Ejemplos de sistemas multi-agente sobre grupos de dispositivos móviles se han implementado para asegurar operaciones en un mercado electrónico [Fischer02] y como asistentes de agenda para viajeros [vanEijk02].

El enfoque multi-agente en el procesamiento de transacciones no es nuevo. En [Vogler98] se presenta una implementación orientada a transacciones distribuidas; su enfoque está centrado en la movilidad de los agentes y está basado en el Servicio CORBA de Transacciones de Objetos para coordinar la transacción global. En el presente trabajo se introduce una implementación basada en un entorno para desarrollo de agentes en lugar de una arquitectura *Broker* de Solicitud de Objetos. Asimismo, no se trata explícitamente el aspecto de movilidad de los agentes, dado que la intención primaria del trabajo es ofrecer una base para el desarrollo de mecanismos más complejos.

En [Nagi01], se trata el problema de robustez en la ejecución de acciones por parte de un Sistema Multi-Agente. La arquitectura está basada en el modelo de transacciones anidadas abiertas. En contraste, el presente trabajo se basa en el modelo de transacciones anidadas cerradas. La diferencia principal radica en la posibilidad de los nodos intermedios y las hojas para confirmar o abortar de forma independiente al nodo raíz (modelo abierto), mientras que en las transacciones cerradas, todos los nodos se apegan a la decisión final tomada por el nodo raíz. En [Straßer98] se presenta otro sistema Multi-Agente orientado a transacciones. Aquí, el objetivo es construir un sistema abierto como tecnología habilitadora para la evolución de un mercado electrónico. En dicho sistema, el aspecto de movilidad se resuelve proporcionando mecanismos de migración. Asimismo, existe una fuerte diferenciación entre agentes de servicio y agentes del usuario. En el presente trabajo, el agente que interactúa con el usuario puede ejecutar las operaciones solicitadas en lugar de solicitar el servicio a otro proveedor.

A pesar de la popularidad que están adquiriendo los dispositivos móviles entre los usuarios, un grupo de trabajo de esta naturaleza (con un propósito en particular) comprende relativamente un número pequeño de participantes. Esta idea se refuerza con la noción de delegar tareas específicas, por ejemplo, el establecimiento de una cita grupal o el compartir datos con el resto del grupo. Por estas razones, el presente trabajo considera un número reducido de peticiones concurrentes sobre recursos compartidos. El mecanismo de control se diseñó con base en un enfoque preventivo, lo cual es popular en aplicaciones de mayor escala [Bernstein97].

En otros trabajos relacionados, se presenta un modelo de simulación para medir el impacto de ciertas características en las transacciones tales como número de hojas y niveles de profundidad con respecto al desempeño del sistema [Hassanein00]. En [Lo92] se introduce el concepto de lista de requerimientos, con el objetivo de proveer tiempos de ejecución cortos y un mejor rendimiento del sistema basándose en intra-transacciones. En [Härder93] se proponen mecanismos de cooperación y comunicación para explotar el paralelismo en intra-transacciones, y se introduce el concepto "Herencia descendente de bloqueos". En [Pitoura95] se proponen soluciones a tópicos de control de concurrencia y recuperación basadas en agentes.

2.4.2 Ventajas de los sistemas multi-agente.

Cada agente proporciona acceso a datos almacenados localmente en los dispositivos y se ejecuta sin dependencia estricta (autonomía relativa) de un sistema central. De esta forma, el proceso de monitorización se vuelve una mezcla de tareas locales y distribuidas. Locales porque cada agente toma decisiones sobre su estado interno. Distribuidas debido a la comunicación constante entre el grupo de agentes, para establecer interacciones que afectan el resultado global de la transacción. De otra forma, basar el resultado en un nodo en particular implica mayor grado de vulnerabilidad en la administración de datos. Las siguientes características también se distinguen al usar un enfoque basado en agentes: mayor concurrencia de transacciones, relajación del nivel de aislamiento, independencia en el *Commit/Abort* de la transacción global, intercambio de mensajes basado en lenguaje de comunicación e información semántica en el contenido del mensaje. Estas ventajas mejoran las posibilidades de incrementar la efectividad de salida de la aplicación.

El árbol de la transacción resultante sigue un esquema jerárquico; sin embargo, cada agente permanece autónomo desde la perspectiva de toma de decisiones. Con excepción del *Commit/Abort* global, el resto de las acciones del grupo no se coordina por medio de una entidad centralizada. Las acciones de los agentes en respuesta al ambiente están estrechamente relacionadas a su estado actual y a sus metas persistentes. Además, los agentes pueden participar como nodos de transacciones concurrentes en una interacción par-a-par, ya sea con su padre o con nodos hijos.

Los SMA remarcan positivamente la concurrencia de transacciones dado que cada agente ejecuta localmente piezas de trabajo en un nodo en particular. La autonomía del agente reduce la dependencia entre transacciones, evitando de esta forma latencia por objetos bloqueados y disminuyendo la posibilidad de abrazos mortales (*deadlocks*) así como abortos en cascada (*cascade-aborts*). Los abrazos mortales se detectan cuando un nodo entra en estado inactivo por esperar la liberación de un objeto que fue bloqueado para su acceso por otro nodo. En turno, éste último, también se encuentra en la espera de que un tercer nodo libere sus candados y así consecutivamente hasta que se forma un ciclo de nodos donde cada cual se encuentra esperando que un vecino libere objetos asegurados. Por otra parte, se ha calculado que la probabilidad de abrazos mortales bajo un número razonable de transacciones es relativamente pequeña [Bernstein97].

Los agentes que soportan Transacciones Anidadas pueden alcanzar sus metas a pesar de las fallas de otros compañeros. Esto se debe al mecanismo de tiempo fuera y a la clasificación de acciones, que se establecieron para reaccionar cuando los

agentes encuentran objetos bloqueados durante su operación. Ambos mecanismos se detallan en la Sección 3.3.4.

La distribución de datos en sistemas manejadores de bases de datos heterogéneos y la distribución de procesos basados en Agentes Autónomos, representan una alternativa robusta para ambientes que se caracterizan por baja estabilidad y fallas frecuentes. Un grupo de dispositivos móviles es distribuido por definición y operado por usuarios autónomos. Además, se caracterizan precisamente por el movimiento como un elemento básico. **Los agentes en dispositivos móviles presentan un entorno adecuado para la implementación de Transacciones Anidadas por las siguientes razones:**

- Las tareas incompletas pueden ser transferidas a otro dispositivo o pueden ser completadas por el dispositivo original una vez que éste regresa a su operación normal.
- Los agentes se comunican efectivamente al tiempo que mantienen un control descentralizado con respecto al estado de la transacción global y las tareas asignadas a cada agente.
- Pueden procesar localmente una transacción mientras se encuentran desconectados de la red de comunicaciones inalámbrica.
- Los programadores son llevados del modelo rígido cliente/servidor, a un modelo par-a-par más flexible, en el cual los agentes se comunican como pares y actúan ya sea como clientes, o servidores de forma simultánea.
- Los agentes de software refuerzan la escalabilidad de aplicaciones, ya que llevan los procesos a la ubicación de los datos, en lugar de llevar los datos al lugar donde se requiere su procesamiento. Ésta es una característica deseable en procesamiento de transacciones anidadas con dispositivos móviles [Gray96].

La arquitectura del sistema multi-agente en el presente trabajo se basa en la Lógica de Interacción, con lo cual se aterriza un mecanismo formal en el campo de las aplicaciones. La ventaja que se obtiene, es el reforzamiento de la robustez del sistema con una teoría subyacente como la estructura clave para modelar interacciones entre agentes.

2.5 Lógica de Interacción

2.5.1 Motivación del modelo formal

La Lógica de Interacción [Alvarado03] [Alvarado02] (LoI) se introdujo como un modelo formal para sistemas multi-agente. Está orientada a tratar con el balance entre el conocimiento y las acciones de los agentes. El aspecto central es que las

acciones individuales de los agentes *interactúan*. Por lo tanto, la representación de las mismas debe hacer explícitas las interacciones y requiere modelar explícitamente distintos procesos. Esto concede importancia a la forma en la cual los agentes desempeñan sus acciones. En este sentido, se proporciona un modelo para acciones sincronizadas, paralelas, secuenciales y concurrentes que son ejecutadas, ya sea por un agente o por un grupo. Se emplea un formalismo modal lógico-temporal para el modelado y representación de acciones.

Para allanar el camino que conecta el modelo formal con su implementación computacional, intencionalmente sólo algunos elementos de la lógica de interacción se tradujeron a código en el desarrollo del presente trabajo. Estrictamente, un monitor de procesamiento de transacciones no requiere ser ensamblado de manera exclusiva a partir de un modelo orientado a agentes; no obstante, éste último es el enfoque experimental que se eligió para el presente trabajo. Por tal razón y de manera natural, surge la necesidad de representar tanto la interacción de acciones de agentes, como la administración de su conocimiento, en lo que a las tareas propias de un monitor de transacciones concierne. Cabe aclarar que por la selección intencional de los elementos aterrizados en código, no se explota en su totalidad el potencial expresivo de una lógica de segundo orden como es la LoI.

2.5.2 Fundamentos de LoI.

Las fórmulas de lógica de orden cero se obtienen en la forma clásica a través de conectores lógicos $\neg$, $\wedge$, $\vee$,$\rightarrow$, $\leftrightarrow$. Las creencias, metas y acciones del agente **i** se representan con las fórmulas modales $B_i\varphi$, $G_i\ \varphi$ y $[do_i\ \alpha]\ \varphi$ "i cree φ", "i tiene la meta φ" e "i ejecuta α para alcanzar φ". El lenguaje modal resultante $\mathcal{L}$ es un subconjunto de la cerradura del lenguaje lógico de orden cero en unión con estos operadores modales.

La sintaxis de la lógica modal define una Estructura de Kripke a la cual se asocia una semántica de mundos posibles. Sea Ag = {1, 2…, n} el conjunto de símbolos que denotan a los agentes y i, j $\in$ Ag, y sea W el conjunto de mundos (interpretación en lógica de primer orden). Un agente i se define por la 3-túpla i = (B_i, G_i, X_i), donde B_i son las creencias del agente (pre-condiciones), G_i son las metas del agente (creencias por alcanzar o pos-condiciones) y X_i son las acciones que el agente puede ejecutar. Sea A = {a_1,...,a_m} el conjunto de acciones atómicas, y sean α y β, las acciones compuestas.

El estado mental de un agente es la 2-túpla de creencias y metas (B_i, G_i). La fórmula φ es una i-creencia $B_i\varphi$ en el mundo s_0, si y sólo si es verdadera en todos los mundos posibles a partir de s_0 por medio de la relación parcial $R_{Bi} \subseteq W \times W$; sea T = {$t_1$, …, t_m} el conjunto de aquellos puntos de tiempo tales que t, t' $\in$ T y $\propto$ un orden

parcial sobre T tal que $t \propto t'$ si y sólo si t' es posterior o igual t. Aunque el factor tiempo no se implementó explícitamente en la aplicación, se requiere para definiciones posteriores en esta misma sección.

2.5.3 Operadores de interacción.

Los cuatro Operadores de Interacción entre agentes se definieron como predicados en los cuales se distinguen dos acciones compuestas a cada lado del operador. Las siguientes observaciones emergen de acuerdo al comportamiento esperado de los agentes durante el procesamiento de una Transacción Anidada. Se muestra además la definición formal.

Sincronía (SYNC).

$$[do_i\ \alpha]\varphi\ sync\ [do_j\ \beta]\psi \qquad (1)$$

Una vez que las operaciones de un nodo padre se han dividido en unidades de trabajo más pequeñas, se espera que el resultado de todas las operaciones se sincronice eventualmente. Es posible sincronizar la ejecución de todos los hijos en cada paso de ejecución. Sin embargo, debido a la naturaleza de las comunicaciones inalámbricas, es sabido que la sincronía se mantiene sólo mientras la conexión está activa. De experiencias anteriores, resulta que el procesamiento de transacciones distribuidas, se asemeja más a un conjunto de operaciones asíncronas, en el cual el principal enfoque está centrado en preservar la consistencia de la base de datos. Esto es, el interés se orienta hacia la sincronía espacial. Aunque se esperaría contar con algunas restricciones temporales, lo más importantes es mantener la sincronía espacial de los objetos en la base de datos. La contraparte del operador SYNC es ASYNC, que manifiesta la asincronía de ejecución de acciones.

Secuencia (SEQ).

$$[do_i\ \alpha]\varphi\ seq\ [do_j\ \beta]\psi \qquad (2)$$

Las operaciones secuenciales en este modelo se realizan en el nivel de hojas. Es en este punto donde cada dispositivo tiene la capacidad de controlar el conjunto de operaciones asignado por el nodo padre. Sin embargo, un nodo padre bajo ciertas circunstancias optaría por ordenar a sus hijos en secuencia en lugar de solicitar su ejecución en paralelo. Por ejemplo, cuando se espera que una sub-transacción confirme exitosamente antes de ejecutar la siguiente sub-transacción dentro de la misma rama. El operador SEQ puede ser combinado con SYNC o ASYNC para

explicitar la sincronía en la secuencia de ejecución de las acciones por parte de los agentes.

Paralelo (PAR).

$$[do_i\ \alpha]\varphi\ \ par\ [do_j\ \beta]\psi \qquad (3)$$

La ejecución en paralelo de tareas emerge como una consecuencia natural de la división de transacciones. Por ejemplo, la Figura 3 (transacción anidada móvil) muestra al Nodo Raíz dividiéndose en sub-nodos. Podría ser la intención del Nodo que se ejecuten sus hijos en paralelo; sin embargo, uno de éstos últimos se vuelve padre también con dos hijos bajo su control. Por lo tanto, existe un total de cuatro hojas ejecutándose en paralelo aunque no era la intención original (desde la perspectiva del nodo raíz) que el árbol se comportara en esta forma. Un nodo padre sólo decide explícitamente cómo ejecutarán los hijos bajo su monitoreo directo. El mecanismo de control desarrollado en el presente trabajo integra comportamientos transaccionales en la ejecución de acciones de los agentes.

Concurrente (CONC).

$$[do_i\ \alpha]\varphi\ conc\ [do_j\ \beta]\psi \qquad (4)$$

La ejecución concurrente se encuentra dentro de los aspectos más importantes en el conjunto de posibles acciones entre los agentes. Sin embargo, debe existir una distinción entre la coordinación de acciones concurrentes de los agentes y la concurrencia en el acceso a datos. La primera hace referencia al comportamiento interno de un sistema multi-agente. Allí, la interacción por sí misma puede requerir que los agentes participantes alcancen un estado concurrente de acuerdo a la definición en [Alvarado02][Alvarado03]. Por otro lado, la concurrencia en el acceso a datos, se refiere al hecho que la consistencia de la base de datos debe ser protegida durante la ejecución de las transacciones. Con el objetivo de mantener esta consistencia, así también la propiedad de aislamiento, debe existir un control más estricto sobre quién y cuándo tendrá acceso a los objetos de la base de datos.

Generalmente, el manejador de la base de datos controla dichos accesos. Sin embargo, el enfoque desarrollado en el presente trabajo se basa en la colaboración de un grupo de agentes para el procesamiento de transacciones, en cuyo contexto, se implementaron tres operadores de la lógica de interacción: PAR, SEQ y ASYNC. En realidad, el operador SEQ se refiere a la versión sincronizada del mismo, esto es SYNC SEQ. En general, el uso de los operadores permite indicar la forma en cómo se liberan las subtransacciones a los nodos hijos desde el punto de vista de un nodo padre.

El tratamiento que se les da a los operandos afectados es de la siguiente forma:

α PAR β. El padre de α y β distribuye de manera simultánea las subtransacciones a los nodos correspondientes que se encargarán de procesarlas.

α SEQ β. El padre de α y β distribuye de forma secuencial las subtransacciones a los nodos correspondientes que se encargarán de procesarlas, esto es, se envía primero el operando α al agente destinatario y a continuación se envía el operando β al nodo hijo que monitorizará el resultado de la subtransacción. La sincronización en secuencia termina en el momento que el nodo padre ha enviado α; posteriormente continua con la transferencia de β. En otras palabras, no se trata de una sincronización de recepción, pues en este último caso, el padre debería asegurarse que el primer hijo ya recibió la subtransacción antes de enviar la siguiente subtransacción al segundo hijo.

α ASYNC β. El operador ASYNC también puede ser expresado como ASYNC SEQ, es decir, las subtransacciones son enviadas desde el padre hacia los hijos en un orden secuencial, sin que esto necesariamente implique una transferencia inmediata después de cada subtransacción, esto es, puede transcurrir cierto tiempo de espera antes de que el nodo padre envíe la subtransacción β a su destinatario correspondiente después de haber enviado la subtransacción α.

2.5.4 Comportamientos JADE y operadores LoI

El monitor de transacciones se implementó en un entorno abierto usando JADE (*Java Agent Development Framework*) como plataforma de agentes [Caire02]. Los agentes emplean ontologías de la plataforma para representar internamente piezas de información [Bellifemine99], mensajes intercambiados y operaciones de control.

La ejecución de tareas por parte de los agentes en la plataforma se implementa y organiza por medio de comportamientos. JADE proporciona dos tipos de comportamientos: simples y compuestos. Cualquiera que sea el tipo, estos comportamientos son ingresados sobre demanda a una cola de procesamiento que es administrada por la plataforma. De esta manera es posible que los agentes cumplan con sus tareas en apego al orden y jerarquía establecidos por el programador.

Un comportamiento simple representa una tarea que puede ser ejecutada una sola vez, ya sea de forma repetitiva hasta que se cumpla cierta condición o bien dentro de un ciclo permanente. La clase base proporcionada por el API es conocida como *SimpleBehaviour*.

Los comportamientos compuestos agrupan combinaciones de otros comportamientos compuestos así como los de tipo simple. Entre las posibilidades de programación que ofrece la plataforma, existen dos casos en particular que se identifican con las expresiones manejadas en la lógica de interacción. Estos son: *SequentialBehaviour* y *ParallelBehaviour*.

Para poder establecer correspondencia entre una expresión de LoI y una cola de comportamientos JADE, aplican las siguientes consideraciones en la versión implementada:

a) SEQ tiene precedencia sobre PAR y ASYNC para atraer a los operandos que están siendo modificados. Esta medida se justifica con el objetivo de aumentar la concurrencia en el acceso a datos por parte de usuarios simultáneos. Con la organización resultante, se obtiene un comportamiento principal de tipo Paralelo dentro del cual existe una lista de comportamientos Simples y Secuenciales. En el caso contrario, que PAR fuera el atractor con precedencia sobre SEQ y ASYNC, el resultado sería una lista de comportamientos *ParallelBehaviour* que, serían despachados en la secuencia definida a través de la expresión.

b) Aquellos operandos afectados por ASYNC deben localizarse en el extremo izquierdo de la lógica de interacción. Nuevamente, aludiendo a la noción de concurrencia, el empleo de ASYNC únicamente en el lado izquierdo de la expresión, permite indicar aquellas subtransacciones que se desea iniciar en secuencia no necesariamente inmediata. El agente es encargado de administrar el momento en que se completa la operación omitiendo las indicaciones explícitas como en el caso de los comportamientos *ParallelBehaviour* y *SequentialBehaviour*. Permitir que ASYNC afecte operandos del lado derecho de la expresión, impactaría la distribución de las subtransacciones al ceder al agente el control sobre cuándo quiera liberarlas hacia el hijo destinatario.

Por ejemplo, si un agente recibe la expresión:

$$\alpha \text{ SEQ } \beta \text{ PAR } \gamma \text{ PAR } \delta \text{ PAR } \lambda \text{ SEQ } \theta \text{ PAR } \rho \text{ ASYNC } \varepsilon \text{ ASYNC } \chi$$

ésta se traduce en el siguiente árbol de comportamientos:

<table>
<tr><td colspan="7">Lista de comportamientos para el agente JADE</td></tr>
<tr><td colspan="5">ParallelBehaviour</td><td rowspan="2">ε</td><td rowspan="2">χ</td></tr>
<tr><td colspan="2">SequentialBehaviour</td><td rowspan="2">γ</td><td rowspan="2">δ</td><td colspan="2">SequentialBehaviour</td><td rowspan="2">ρ</td></tr>
<tr><td>α</td><td>β</td><td>λ</td><td>θ</td></tr>
</table>

donde cada operando está representado por un comportamiento de tipo *SimpleBehaviour*.

2.5.5 Anidamiento de subtransacciones.

En una expresión de lógica de interacción los paréntesis agrupan subtransacciones dando lugar a una rama anidada dentro del árbol. Por ejemplo, la siguiente expresión

$$(\alpha \text{ SEQ } \beta) \text{ PAR } \gamma \text{ PAR } (\lambda \text{ PAR } (\delta \text{ SEQ } \theta))$$

es enviada al nodo raíz como

$$\alpha' \text{ PAR } \gamma \text{ PAR } \lambda'$$

en donde

$$\alpha' = \alpha \text{ SEQ } \beta, \lambda' = \lambda \text{ PAR } \delta' \text{ y } \delta' = \delta \text{ SEQ } \theta.$$

Es decir, el nodo raíz de la transacción anidada, divide la transacción original en tres subtransacciones. Dos de ellas, α'y λ' a su vez, están compuestas por nodos anidados, y en el caso de λ', contiene un nivel más de profundidad representado por δ'. Los operadores PAR y SEQ indican al padre de la transacción o subtransacción, la forma en cómo debe enviar las piezas de trabajo a sus respectivos hijos. El árbol final una vez completada la interpretación de la expresión LoI se ilustra en la Figura 5.

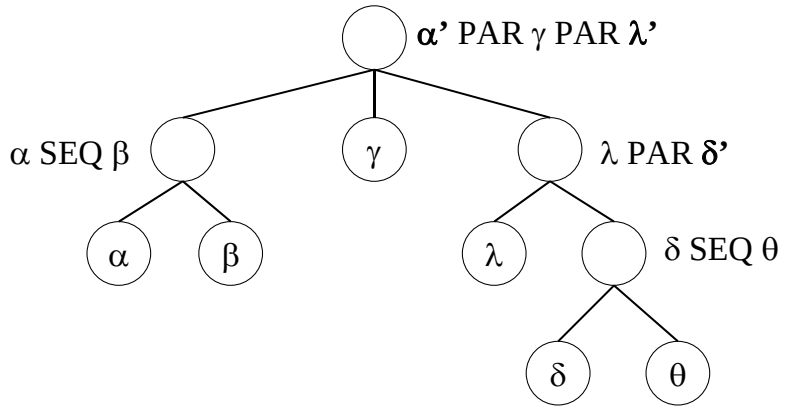


Figura 5. Árbol de transacciones generado a partir de la expresión LoI que recibe el nodo raíz

Un operando compuesto en la lógica de interacción, por ejemplo φ o θ, representa a su vez una secuencia de instrucciones SQL. Éstas últimas son ejecutadas por las hojas sobre la base de datos incrustada. De manera formal, un operando φ equivale

$$a \text{ SEQ } b \text{ SEQ } \ldots \text{ SEQ } n,$$

donde a,b,…,n son instrucciones SELECT, INSERT, UPDATE, como los siguientes ejemplos: “SELECT NODE7.STRING FROM NODE7 WHERE ID=1“, “UPDATE NODE7 SET DATE = ‘20050505’ WHERE ID=1“. Es importante mencionar que en este caso, la lista de operaciones representa una transacción plana, lo cual se corresponde a la definición anidada, en la cual, las hojas procesan transacciones planas. Cabe mencionar que en los operandos compuestos (α, β, etc.), se encuentra implícito el operador SEQ de la lógica de interacción. La diferencia con respecto al operador que se maneja durante la expansión del árbol, radica en que en las hojas, SEQ define el orden secuencial en que deben ser procesadas las operaciones de lectura o escritura. En cambio, durante la interpretación de las expresiones LOI, SEQ indica el orden en que un padre envía las subtransacciones entre sus nodos hijo.

3 Monitor de procesamiento de transacciones

En este capítulo se presenta y discute la implementación del monitor desde la perspectiva de la arquitectura por tres capas: Presentación, Flujo de Trabajo y Base de Datos. En particular, la segunda capa introduce los mecanismos para el control de concurrencia: el protocolo de compromiso a dos fases y el administrador de candados.

De acuerdo al modelo JADE para diseño y programación de sistemas multi-agente, las tareas que se ejecutan deben estar organizadas por medio de comportamientos. En el monitor de procesamiento de transacciones, el comportamiento principal denominado *CommandListener* se encarga de procesar los mensajes del agente en cuestión. Es de tipo *SimpleBehaviour*, esto es, toda su funcionalidad está encapsulada en el método *action()* y además el programador es responsable de indicar en qué momento el comportamiento debe considerarse como finalizado. En el caso del monitor, esta situación se presenta sólo cuando el agente abandona la plataforma y por lo tanto, todos sus comportamientos activos son terminados por JADE.

Otra de las características que ofrece JADE y que se usó en la implementación del monitor, es el manejo de ontologías definidas por el usuario. De esta manera se representa las expresiones de lógica de interacción como objetos Java que son procesados de manera interna por los agentes. Adicionalmente, por medio de la ontología se mantiene una representación común de la información que es transmitida dentro del sistema multi-agente. Por medio de los mensajes, se solicita la ejecución de acciones desde un nodo padre hacia sus hijos; asimismo, el resultado de haber ejecutado cierta acción, es reportado del nodo hijo hacia el padre con información que se encapsula en objetos de la ontología definida para tales fines.

3.1 Arquitectura

3.1.1 Capas del monitor

Un Monitor de Procesamiento de Transacciones se concibe como el *software* a cargo de administrar peticiones simples de los usuarios que serán escaladas sobre un sistema distribuido [Bernstein97]. También es responsable de preservar las

propiedades ACID durante la ejecución concurrente de transacciones. En pocas palabras, el monitor:

- recibe una petición,
- la traduce a un lenguaje comprensible por el resto del sistema,
- marca el inicio de la transacción,
- controla las operaciones *commit* o *abort* y finalmente
- reporta el resultado al usuario.

De acuerdo a [Orfali95], los Monitores de Procesamiento de Transacciones que contemplen el modelo Anidado como la base para representar aplicaciones de misión crítica, reflejarán en mayor medida la naturaleza de los procesos de negocios.

La estructura general de un monitor de Procesamiento de Transaccciones se identifica por tres capas:

1. *Presentación*. Recibe las instrucciones del usuario y las traduce a un lenguaje comprensible por el sistema. Estas instrucciones se transfieren a la capa de flujo de trabajo. Posteriormente, el resultado es presentado de regreso al usuario.
2. *Flujo de trabajo*. Como su nombre lo menciona, esta capa es responsable de enrutar, administrar y responder las peticiones recibidas desde la capa de presentación. Las instrucciones se envían a la tercera capa y los resultados se regresan a la primera.
3. *Base de Datos*. El acceso a los objetos se completa en este nivel. Los resultados, ya sean exitosos o no, se informan a la capa anterior. En el presente trabajo, la implementación se basa en una base de datos incrustada en cada dispositivo: *Pointbase Micro Edition* [Pointbase04]. Dado que el entorno de desarrollo de agentes se fundamenta en el lenguaje de programación Java, era necesario contar con una base de datos compatible que pudiera ejecutarse en dispositivos móviles.

3.1.2 Clases ontológicas

El manejo de ontologías en los sistemas de información, en particular aquellos desarrollos en el área de Inteligencia Artificial, no está exento de acuerdos y desacuerdos sobre las definiciones, interpretaciones e implementaciones posibles. Para fines prácticos, la plataforma de desarrollo JADE considera una ontología como el conjunto los elementos en un dominio de discurso. Este conjunto debe estar soportado por un mecanismo que permite traducir e interpretar entre piezas de información y objetos manipulables desde el punto de vista del desarrollador.

Esta traducción automática entre expresiones de contenido y objetos del lenguaje de programación, es una ventaja particular en la representación de información con estructura compleja que deben ser transferidas entre agentes. De esta forma se logra un acuerdo en la forma en que los agentes deben interpretar las piezas de información compartidas durante el monitoreo del procesamiento de transacciones.

De acuerdo a las herramientas proporcionadas por JADE, es necesario definir una clase ontológica que represente a cada elemento perteneciente al dominio de la aplicación. En el sistema multi-agente se definieron tres clases de este tipo, dos clases para representar expresiones de lógica de interacción y una para representar instrucciones de control que un agente debe ejecutar por solicitud de otro. El resultado de la ejecución también se reporta usando clases de este tercer tipo.

En JADE, una clase ontológica se compone de ranuras que pueden ser tipos básicos de datos (entero, cadena de caracteres, flotante, booleano), agregados (listas) u objetos compuestos. El monitor desarrollado se apoya en:

Clase *Instructions*:
Es la representación de un operando compuesto: α, β ,… Contiene una lista de cadenas de texto para almacenar las instrucciones SQL que se ejecutarán en las hojas. Además tiene una ranura de tipo entero que indica el lugar asignado al operando en la expresión anidada a la cual pertenece:

- Una lista de cadenas, cada elemento de la lista es una instrucción SQL, por ejemplo: "SELECT NODE7.STRING FROM NODE7 WHERE ID=1".
- El número asignado a este paquete de instrucciones para ser enviado del nodo padre al nodo hijo.

Un objeto *Instructions* se emplea cuando un nodo intermedio debe mandar un conjunto de instrucciones SQL a un nodo hoja. Dado que implementa la interfaz *Concept*, no puede ser incrustado directamente en el contenido de un mensaje, por lo que se envía a través de un objeto *Command* explicado más adelante.

Clase *Simpleoper*.
Se emplea para almacenar el contenido de una expresión de lógica de interacción, en otras palabras, es la representación de árboles distribuidos y anidados.
Cuenta con las siguientes ranuras:

- Lista de cadenas, cada elemento de la lista es un operador de la LoI.
- Lista de objetos *Simpleoper*, expresiones anidadas de la lógica de interacción.
- Lista de objetos *Instructions*, paquetes de operaciones que serán enviados desde el nodo actual hacia un nodo hoja.

- El número asignado a esta expresión de acuerdo al orden en que será procesado con respecto a las demás piezas de trabajo que el agente distribuirá entre sus nodos hijos.

Un ejemplo de representación de una transacción distribuida se muestra en la Figura 6,

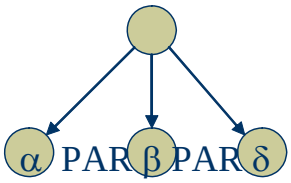


Figura 6. Expresión LoI para una transacción distribuida

donde la lista de operadores contiene los elementos {PAR, PAR}, la lista de objetos *Simpleoper* se encuentra vacía mientras que la de *Instructions* contiene los tres objetos {α,β,δ}.

En el caso de una transacción anidada (Figura 7)

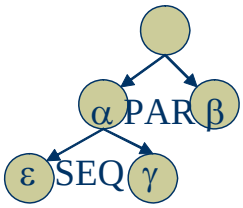


Figura 7. Expresiones LoI para una transacción anidada

donde α es un objeto de tipo *Simpleoper* que cuenta con: una lista de operadores {SEQ}, una lista de objetos *Instructions* {ε,γ} y su respectiva lista de objetos *Simpleoper* está vacía. Sin embargo, α a su vez forma parte de una transacción anidada donde la lista de operadores es {PAR}, la lista de objetos *Simpleoper* contiene a {α} y la lista de objetos *Instructions* almacena únicamente a {β}.

Al igual que *Instructions*, esta clase implementa la interfaz *Concept* de JADE por lo que debe ser incrustado en la ranura correspondiente de un objeto *Command* para que un agente pueda mandar la información a otro agente.

Interfaz *Vocabulary*.
Comprende un conjunto de constantes para representar instrucciones, estados y niveles de prioridad en el monitoreo y procesamiento de transacciones. Las constantes se agrupan en categorías.

Tipo de instrucción/informe. Indican la operación o que un agente espera sea ejecutada por el destinatario del mensaje, o bien el resultado de la operación solicitada.

OPEN_TRANSACTION. Su uso está restringido al agente cliente y al agente que administra el nodo raíz de una transacción.

OPEN_SUBTRANSACTION. Indica al receptor que debe unirse a una transacción padre y revisar además expandir el árbol de subtransacciones o, en el caso de las hojas, ejecutar el conjunto de instrucciones SQL.

ABORT_TRANSACTION. Cancelar todas las operaciones pendientes relacionadas con la transacción indicada y eliminar la información registrada hasta el momento.

JOIN. Este mensaje lo recibe un nodo padre como notificación de que el nodo hijo ha registrado la subtransacción exitosamente y se integra al árbol de procesamiento.

CAN_COMMIT. Un nodo padre pregunta al hijo si está listo para confirmar el resultado de la transacción.

TWOPC_YES/TWOPC_NO. Respuesta a la petición CAN_COMMIT, el agente ha encontrado que la transacción está lista o no para confirmar.

DO_COMMIT/DO_ABORT. Un nodo padre indica al hijo que confirme o aborte la transacción ya que esa fue la decisión final de la raíz.

Clase *Command*.

Es el elemento de ontología principal dado que encapsula información sobre un comando que un agente debe ejecutar por solicitud de otro agente, o bien, información sobre el resultado de haber ejecutado una acción previamente solicitada. Las ranuras que lo componen son:

- Tipo. Identificador numérico del comando, instrucción u operación que debe ser ejecutada, o bien del resultado que se está reportando. Los códigos se encuentran definidos en el archivo de vocabulario anteriormente mostrado.
- Identificador de la transacción. Se trata de una ranura opcional, se emplea en aquellos casos donde el destinatario requiere saber cuál es la transacción en proceso.
- Objeto *Simpleoper*. En esta ranura se especifica una expresión de lógica de interacción codificada de acuerdo a los elementos de ontología del sistema multi-agente. Se emplea cuando un nodo raíz o algún nodo intermedio envía una estructura jerárquica a un hijo, para que a su vez, este último, decodifique la expresión y divida las piezas de trabajo entre más participantes. Es una ranura opcional, ya que en su lugar, puede ser empleada la ranura *Instructions*.
- Objeto *Instructions*. Aquí se especifica un paquete de instrucciones que serán enviados a un nodo hoja. Es decir, el receptor ya no expande la profundidad

del árbol de transacciones, sino que recibe directamente una lista de operaciones SQL que deberá ejecutar.
- Identificador de subtransacción. En ciertas operaciones, se debe comunicar este identificador, mismo que es generado por un nodo hijo que está reportando información hacia su padre.
- Hora de inicio de la transacción. Es una estampa de tiempo generada por el nodo raíz de una transacción y que debe ser transmitida a todos los participantes.
- Bandera de autorización para uso del tiempo fuera (*timeout*). En algunos casos, de acuerdo a la tabla de clasificación de acciones que se describe más adelante, los nodos hoja pueden hacer uso de un tiempo fuera – actualmente definido en diez segundos- esta situación se les informa de manera explícita para que el agente sepa si debe realizar o no, nuevos intentos de ejecución de las instrucciones SQL que le fueron asignadas.

Por las restricciones de la máquina virtual Java para dispositivos móviles, la plataforma JADE proporciona una representación alternativa del contenido de los mensajes. En lugar de consistir de cadenas, que es como lo especifica el estándar FIPA, se emplean secuencias de *bytes*. De esta forma, el paso de mensajes se adapta a las restricciones en capacidad de los dispositivos móviles.

Como se mencionó anteriormente, para que un agente solicite la ejecución de alguna acción o bien reporte el resultado de haberla ejecutado, envía un mensaje que contiene un objeto *Command* con la información que corresponda. El mensaje es procesado a través del comportamiento *CommandListener*. Este comportamiento es usado en su mayor parte por la capa de flujo de trabajo y en menor medida por las capas de presentación y de base de datos.

3.1.3 Estructura de los agentes

Los recursos disponibles en dispositivos móviles tanto en capacidad de cómputo como de almacenamiento, representan una característica a tomar en cuenta en el diseño de aplicaciones. En el sistema multi-agente, la funcionalidad esperada del monitor se distribuye en las operaciones que realizan de manera individual cada agente que participa en el procesamiento de transacciones.

De acuerdo a las particularidades del entorno de desarrollo seleccionado, las diferentes actividades relacionadas con monitoreo de las operaciones, se implementa por medio de comportamientos.

Las interacciones entre el grupo de agentes se logran por medio del paso de mensajes; básicamente se clasifican en dos tipos: petición e informe. El primero de

ellos comprende tanto aquellas solicitudes para ejecutar alguna operación o bien para reportar datos relacionados con el estado del agente que recibe el mensaje. El segundo tipo se emplea para enviar información, ya sea porque fue solicitada por un mensaje de petición, o bien por ciertas condiciones del estado actual de un agente, que disparan el envío.

El comportamiento JADE por medio del cual se reciben los mensajes y se procesan en consecuencia es *CommandListener*. Las acciones que se ejecutan de acuerdo a los mensajes recibidos son: abrir una transacción, abortar una transacción, abrir una subtransacción, disparar el inicio del protocolo de compromiso atómico. En el rol de nodo raíz o nodo padre, en este comportamiento también se administran las respuestas de los hijos. Las hojas, por su parte, en este comportamiento realizan el primer intento de ejecución de las instrucciones SQL sobre la Base de Datos incrustada.

Existen además, dos comportamientos adicionales. *CommandDispatcher*, que se emplea para enviar las piezas de trabajo a los hijos de un nodo y soporta la construcción del árbol de subtransacciones de acuerdo a lo explicado en las secciones 2.5.2 y 2.5.3. *TimeOutManager* es otro comportamiento que se emplea sobre demanda y sólo en el nivel de hoja; su funcionalidad permite notificar al agente que se ha vencido el tiempo fuera para cierta subtransacción y por ello debe notificar el resultado a su nodo padre.

Los agentes se apoyan en clases auxiliares para administrar información del monitoreo, de la ejecución de transacciones, así como del control de concurrencia. Éstas son:

- *xacEntry*, *subXacEntry*, *leafEntry*: almacenan información específica sobre transacciones y subtransacciones.
- *SQLParser*: administra la introspección de instrucciones SQL.
- *SentenceDescription*: separa una instrucción SQL en sus campos constituyentes.
- *Lock*: mantiene una lista de los usuarios de un objeto de datos, rechaza o concede la adquisición de un candado.
- *LockManager*: Establece o libera candados bajo petición de una hoja que se encuentra procesando instrucciones SQL.
- *statLogger*: almacena en un archivo de texto números relacionados con la actividad de procesamiento y monitoreo del agente.

3.2 Capa de Presentación

Representa el medio de interacción entre el usuario y el monitor. También se encarga de traducirlas en el lenguaje de representación interna. Como se ha

mencionado, la implementación actual se basa en expresiones de lógica de interacción expresadas con ayuda de clases ontológicas JADE.

Las características en diseño de la interfaz se encuentran estrechamente relacionadas con el objetivo de la aplicación. En el trabajo desarrollado, las pantallas de entrada y salida de datos, demuestran una forma posible en que se recibe información del usuario por medio de un agente cliente. Éste se encarga de generar el objeto *Simpleoper* correspondiente que será enviado al dispositivo raíz de la transacción anidada. Allí, un segundo agente se encargará de descomponer la expresión de lógica de interacción en lo que ya forma parte de la capa de flujo de trabajo.

A través del agente cliente, el usuario tiene la opción de abortar manualmente la transacción o bien esperar el resultado final. En este segundo escenario, existen tres posibilidades para la transacción: terminó exitosamente, se ejecutaron algunas operaciones pero otras fallaron o falló toda la transacción. En los últimos dos casos, el usuario tiene la última palabra sobre si desea que se reintente la ejecución de aquellas piezas de trabajo fallidas.

La Figura 8 ilustra la parte visible de la capa de presentación para el usuario en el ejemplo desarrollado.

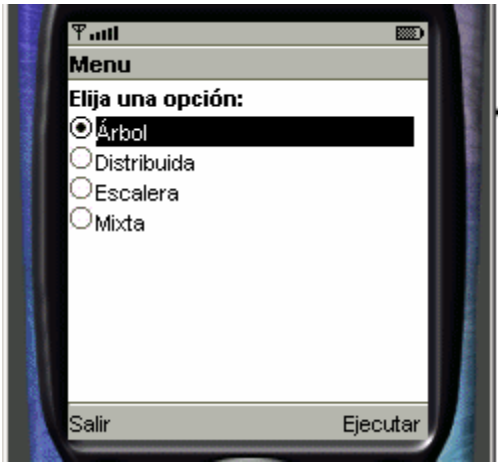


Figura 8. Interfaz de usuario del monitor

La representación de una transacción anidada y su correspondiente procesamiento, que incluye la forma en cómo se construye el árbol, puede apreciarse de dos maneras. La primera se trata de una composición dinámica en la cual la transacción, inicialmente, es un conjunto de operaciones que bien podría asemejar una transacción plana. El nodo raíz la recibe e internamente determina en cuántas piezas de trabajo debe dividir al conjunto original; asimismo, también decide la forma en que serán enviadas a los nodos hijo, ya sea en secuencia, paralelo o

secuencia asíncrona. El proceso se repite con cada uno de los nodos hijo, quienes reciben un subconjunto de instrucciones de la transacción original, determinan en cuántas partes dividirlo y reparten, a su vez entre nodos hijos las piezas generadas. Siguiendo este tipo de composición, un nodo también puede adoptar el rol de hoja en el árbol de subtransacciones. La Figura 9 muestra un ejemplo de composición dinámica de una transacción anidada. En ella, el nodo raíz recibe la transacción α y decide repartir la carga de trabajo entre tres nodos; el signo '?' indica la selección de operadores entre SEQ, PAR y ASYNC. A continuación, β_1 y β_3 participan en la transacción como hojas, es decir, intentarán ejecutar las instrucciones SQL sobre su copia local de datos. En contraste, β_2 divide el subconjunto de instrucciones recibidas en dos partes que serán enviadas a los respectivos hijos. Finalmente, γ_1 y γ_2 se unen a la transacción como hojas del árbol con lo cual queda configurada la transacción anidada.

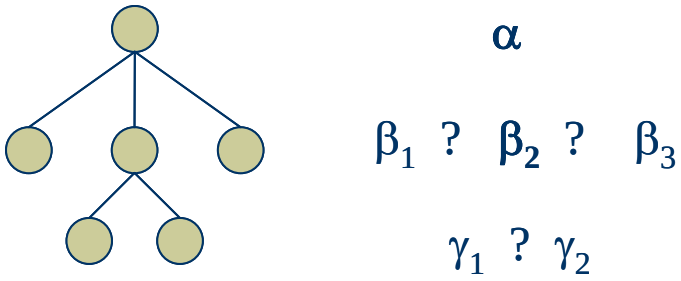


Figura 9. Composición dinámica de una transacción anidada: los agentes deciden en tiempo de ejecución, con qué operadores (marcas '?') se enviarán las subtransacciones a sus nodos hijos.

Figura 10. Composición predefinida de una transacción anidada: los agentes reciben desde el nivel superior (o el cliente en el caso del nodo raíz), los operadores de la expresión para controlar el envío de las subtransacciones a sus respectivos hijos.

El monitor de procesamiento de transacciones emplea una composición predefinida, misma que permite la representación explícita con expresiones de lógica de interacción. En esta modalidad, el nodo raíz recibe de la aplicación cliente, indicaciones sobre el número de nodos que se requiere añadir al árbol, así como el orden (en secuencia, paralelo, secuencia asíncrona) en que se les enviará su respectivo subconjunto de instrucciones. En la Figura 10 se muestra un ejemplo

donde la expresión α SEQ β' PAR δ es recibida por el nodo raíz, quien interpreta que la transacción original debe dividirse en tres piezas de trabajo que serán enviadas a igual número de hijos. En la estructura predefinida de β', el nodo coordinará la ejecución de las subtransacciones ε y γ.

3.3 Capa de Flujo de Trabajo

3.3.1 Expansión del árbol.

Las actividades del nodo raíz dan inicio a partir de la llegada de la transacción que envió un agente de usuario. Los datos están encapsulados en un objeto de tipo *Simpleoper*. En la siguiente etapa se registran los datos necesarios para monitorizar el procesamiento de la transacción, estos son:

a) hora de inicio (actualmente controlada por el nodo raíz),
b) identificador único: es una llave compuesta por el nombre corto JADE del agente, la letra "T" y un número secuencial, por ejemplo "rootT3",
c) dirección del nodo padre: aunque formalmente un nodo raíz no tiene padre, se registra la dirección del agente usuario a quien le será enviado el resultado de la transacción,
d) árbol de subtransacciones: se genera mediante la interpretación del contenido del objeto en correspondencia a los elementos de LoI involucrados. El algoritmo de composición del árbol se detalla en la sección 3.3.2.

En la tercera etapa (recursiva) se completa el envío físico de las subtransacciones a sus respectivos destinatarios, que puede ser un nodo hoja o bien un nodo intermedio que coordinará la ejecución de subtransacciones adicionales; en este segundo caso, el agente procede conforme lo indicado para la segunda etapa, con la salvedad de que la hora de inicio se consulta directamente del mensaje recibido.

Tal como lo realizara su nodo padre, en el nodo actual se completa el envío físico de las piezas de trabajo identificadas. Eventualmente, éstas llegan hasta el nivel más profundo del árbol donde se localizan los nodos hoja. Aquí se registran los datos necesarios de la transacción; en contraparte a lo que ocurre con nodos superiores, una hoja extrae el objeto *Instructions* del mensaje recibido e intenta ejecutar la lista de instrucciones SQL sobre su copia local de datos. Ya sea que tenga éxito o no en este primer intento, la hoja responde con una notificación JOIN de ingreso al árbol de la transacción. Cuando el nodo padre ha recolectado todas las respuestas de sus hijos, genera su propia notificación para enviarla a su antecesor. Una vez que el nodo raíz ha obtenido todas las notificaciones de los nodos inferiores, se da inicio al protocolo de compromiso atómico.

3.3.2 Algoritmo del intérprete LoI

Al despliegue de la transacción se genera un árbol de subtransacciones bajo la supervisión de cada nodo participante, excepto las hojas. Físicamente, dicho árbol está representado por comportamientos JADE. Éste se construye a partir de la información recibida en un mensaje ya sea de tipo OPEN_TRANSACTION para el nodo raíz, o OPEN_SUBTRANSACTION cuando se trata de nodos intermedios. Las operaciones que integran el intérprete LoI se encargan de:

1. Ordenar los operandos: originalmente, se tienen dos listas de subtransacciones; la primera contiene los datos que serán enviados a hijos quienes a su vez coordinarán el procesamiento de más hijos; la segunda lista contiene datos que serán enviados a nodos hoja bajo la coordinación directa del nodo actual (en expansión). Cada uno de estos elementos se etiqueta según el orden que le corresponde en la representación del árbol de la transacción. Un ejemplo se muestra en la Figura 11.a donde un nodo raíz recibe una transacción que consta de cuatro subtransacciones; a su vez, la segunda de ellas es una transacción compuesta de dos elementos. Durante la reconstrucción del árbol (comportamientos JADE) es posible ubicar el nodo β en el sitio donde le corresponde para que se mantenga la estructura deseada.
2. Localizar al proveedor de datos. Para cada subtransacción, el nodo evalúa quién de los agentes que participan en el grupo de monitoreo posee los datos que requieren ser consultados o modificados. Este proceso se compone de dos actividades: realizar una introspección de las operaciones que componen la subtransacción y consultar entre los demás agentes disponibles, quién tiene acceso a los datos requeridos. Actualmente, con fines de pruebas, los nombres de los agentes incluyen un número serial que permite identificarlos; así, el proveedor de datos para cada subtransacción del árbol que se está procesando, es el siguiente agente en la numeración. Por ejemplo, si el agente actual se llama *node2* y debe distribuir 3 subtransacciones, entonces, los destinatarios serán: *node3*, *node4* y *node5*. En el caso del agente con el nombre *nodeN*, tal que es *N* el último nodo, su sucesor es *node1*.
3. Asignar a la subtransacción un nivel de prioridad entre: crítico, obligatorio fuerte, obligatorio débil, opcional. El nivel seleccionado, también sirve para determinar si el nodo receptor (en caso de que se trate de una hoja) está autorizado para hacer uso de un período de tiempo durante el cual intentará ejecutar su lista de operaciones en caso de que surjan conflictos por acceso concurrente.
4. Crear los objetos *CommandDispatcher* (de tipo comportamiento simple) que se encargarán de completar físicamente la transferencia de la subtransacción al nodo destinatario.

5. Agregar un comportamiento JADE de tipo *ParallelBehaviour* a la cola del agente en donde se incluyen los operandos modificables por PAR tales que no están atraídos por SEQ.
6. Crear un comportamiento *SequentialBehaviour* para cada grupo de operandos que son modificados por SEQ, y agregarlos al *ParallelBehaviour* principal.
7. Por último, agregar directamente a la lista de comportamientos del agente, todos aquellos operandos modificados por ASYNC

La Figura 11.a muestra un ejemplo donde el nodo raíz recibe el objeto *Simpleoper* del agente cliente con los siguientes datos: lista de operadores {SEQ, PAR, ASYNC}, transacciones planas {α,δ,ε}, transacciones compuestas {β}. El objetivo es reconstruir la expresión original α SEQ β PAR δ ASYNC ε. El primer paso consiste en crear una lista temporal con los elementos en el orden que le corresponden {α, β, δ, ε}. A continuación se crea un comportamiento *CommandDispatcher* (CD1 a CD4) para cada subtransacción respectivamentes. Se crea además una instancia de *ParallelBehaviour*. Dado que la lista de operadores fue creada en orden, el agente puede evaluar qué operandos están afectados por PAR, SEQ o ASYNC. De acuerdo a las reglas del intérprete, la configuración de los comportamientos que serán agregados a la cola del agente se muestra en la Figura 11.b.

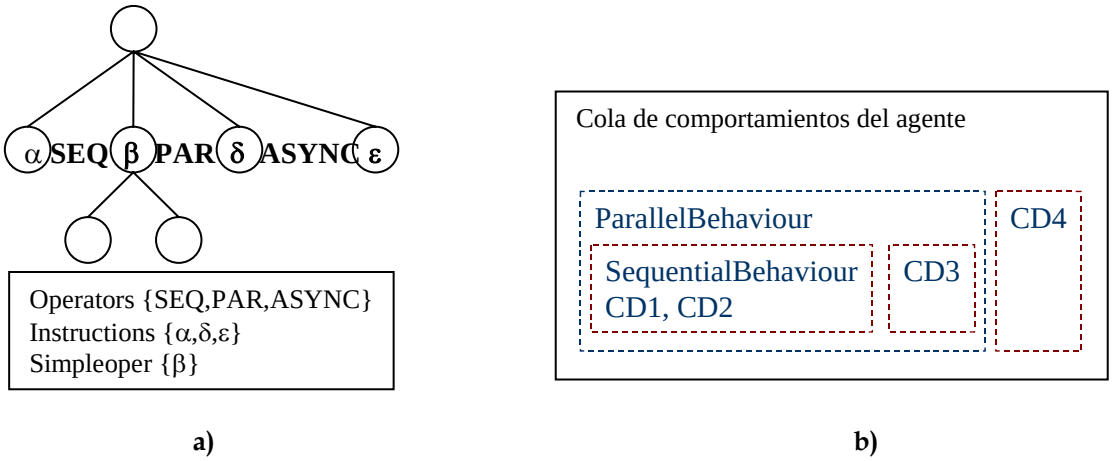


Figura 11. Expresión LoI y cola de comportamientos resultante: a) expresión de LoI para una transacción anidada, árbol correspondiente y contenido de los objetos en el mensaje que recibe el nodo raíz; **b)** cola de comportamientos generada a partir de la expresión LoI.

3.3.3 Protocolo de Compromiso Atómico

Por medio de este mecanismo, un conjunto de agentes participantes se unen a la transacción (iniciada por algún usuario a través de un nodo raíz) y reciben un subconjunto de las operaciones de la transacción completa. Los nodos descendientes pueden unirse a cada participante hasta que se completa un árbol. Una vez que el árbol está listo:

Fase 1. El nodo raíz libera una petición *canCommit* a todos sus hijos. Esta petición se transfiere jerárquicamente hacia abajo y eventualmente alcanza el nivel de las hojas. Desde aquí, los nodos reportan SI o NO hacia arriba hasta que la información alcanza al nodo raíz. Los nodos intermedios toman decisiones provisionales de acuerdo a los resultados reportados por sus hijos.

Fase 2. Con las respuestas obtenidas, el nodo raíz decide si es posible confirmar con las ramas exitosas o debe abortarse la transacción. Este resultado se libera en forma similar a la petición de la fase 1. Los nodos con un ancestro abortado, deben abortar también.

El proceso anterior se ilustra en la Figura 12. Allí, el nodo raíz decide confirmar a pesar de la falla de T_1. Dado que T_{12} es un hijo T_1, debe abortar en consecuencia.

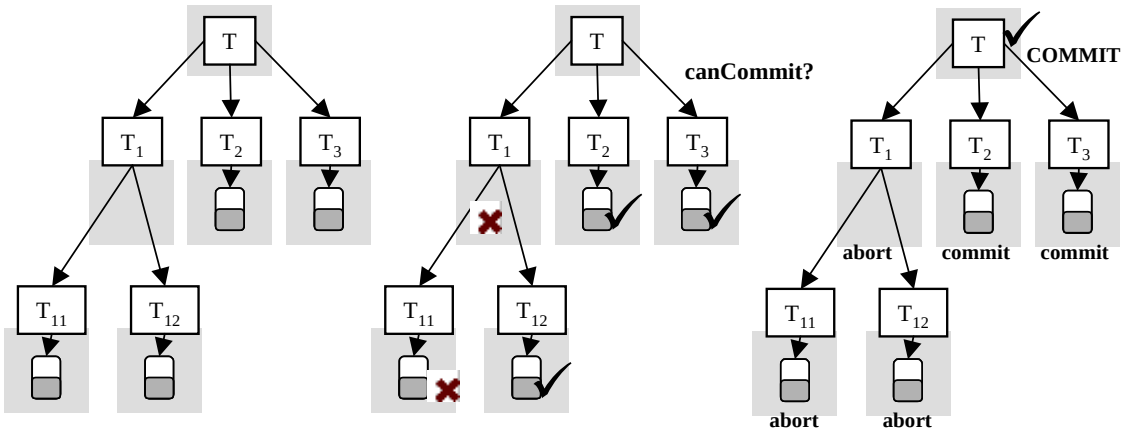


Figura 12. Protocolo de Compromiso a dos Fases para Transacciones Anidadas

El mensaje SI/NO que un nodo envía a su padre como respuesta a la petición *CanCommit*, va acompañado de las piezas de trabajo fallidas bajo la administración de dicho nodo. De esta forma, eventualmente el nodo raíz, obtiene la lista de todas aquellas operaciones que no pudieron completarse. Al informar la decisión final del nodo raíz hacia el agente del usuario, también envía el subconjunto de operaciones para que éste último decida si deben realizarse nuevos intentos por medio de una nueva transacción.

3.3.4 Control de Concurrencia.

Durante el procesamiento de transacciones, los agentes pueden encontrar que algunos datos ya han sido bloqueados debido al acceso concurrente por parte de otros clientes. Para tratar con este tipo de situaciones, se implementó un mecanismo de control; se trata de un comportamiento incrustado en cada uno de

los agentes que participan en la transacción. Los nodos del árbol pueden ser clasificados como alguno de los siguientes tipos:

1. Crítico,
2. Obligatorio Fuerte,
3. Obligatorio Débil,
4. Opcional.

La columna de descripción en la Tabla 1 muestra qué se debe hacer en caso de que un nodo falle al ejecutar las operaciones que le fueron asignadas.

Tabla 1. Clasificación de nodos para control de concurrencia en el procesamiento de transacciones

Tipo	Procedimiento
Crítico	Se aborta la transacción.
Obligatorio Fuerte	Se programan nuevos intentos hasta que la transacción pueda completarse o expira después del tiempo fuera; en este segundo caso, se aborta la transacción.
Obligatorio Débil	La transacción se envía de regreso al usuario en espera de nuevas instrucciones.
Opcional	Se programan nuevos intentos hasta que la transacción pueda completarse o expira después del tiempo fuera y la transacción se envía de regreso al usuario como en el caso anterior.

Las siguientes reglas también aplican:

- Si una transacción Crítica/Obligatoria Fuerte falla, toda la rama se considera no exitosa. Eso es, todos los nodos en el mismo nivel que pertenecen al mismo padre, también abortan.
- Si para cierto nodo, todos sus hijos están clasificados como Obligatorio Débil u Opcionales, sólo se requiere el éxito de un hijo para considerar toda la rama como exitosa.

La clasificación anterior considera únicamente dos casos para el éxito de la ejecución de un nodo hijo: que éste falla o que éste tiene éxito. Concede asimismo, un criterio para determinar si es imperativo intentar nuevamente la ejecución de operaciones no exitosas. La tabla (y el mecanismo en general) pueden ser extendidos para considerar casos donde se requiera el éxito específico de un número *n* de nodos hijos en la ejecución de sus transacciones.

Al etiquetar a sus agentes hijos en alguno de los tipos arriba enumerados, un agente-nodo del árbol de la transacción, se habilita con elementos para tomar decisiones como parte del protocolo de compromiso atómico. Ejemplos de

situaciones problemáticas se tienen cuando un nodo intenta acceder a un objeto que ya se encuentra bloqueado e incluso cuando se pierde la comunicación con los demás agentes. Como alternativa de operación, se puede organizar un nuevo grupo de nodos descendientes que reemplace a la rama original.

Si las operaciones fallidas no son obligatorias, es posible programar nuevos intentos por medio de una nueva transacción. En este punto se introduce la noción de *persistencia* de acciones en el sentido que, el agente intentará completar el conjunto de instrucciones que se le asignó originalmente – siempre que esto sea posible.

Dado que el colapso de los *hosts* móviles y la interrupción en las comunicaciones son eventos que pueden presentarse en un ambiente móvil, el mecanismo propuesto puede operar como base para controles de bitácora y recuperación.

Control de concurrencia basado en candados.

El objetivo principal del mecanismo es tratar con conflictos que surgen cuando dos o más nodos (ya sea de la misma transacción o de diferentes transacciones) tratan de acceder al mismo objeto. Tomando en cuenta las condiciones que caracterizan la operación de dispositivos en ambientes móviles, el control debe asimismo considerar las restricciones en tiempo y recursos de cómputo disponibles.

El procedimiento de operación se ilustra con ayuda de la Figura 13. Allí, la subtransacción T_{13} de T_1 ya ha bloqueado un objeto de la base de datos al tiempo que una segunda subtransacción T_{211} de T_{21} trata de acceder al mismo objeto.

En cada dispositivo móvil reside un agente con un micro Manejador de Candados incrustado. Éste cuenta con una tabla de información sobre los objetos y los bloqueos actuales en el dispositivo anfitrión. Los candados se mantienen durante el tiempo de vida de la transacción y son inmediatamente liberados después que se recibe el *Abort* o *Commit* definitivo desde el nodo raíz.

Suponiendo que existe una petición de bloqueo compartido para acceso de lectura por parte de T_{13}, el Administrador registra qué transacción y objeto están vinculados durante el tiempo necesario hasta su conclusión. Si una segunda transacción, T_{211}, desea leer el mismo objeto, el Administrador autoriza el acceso compartido dado que no existen conflictos entre ambas peticiones.

Los problemas se presentan cuando al menos una de las transacciones solicita acceso exclusivo dado que requiere escribir sobre el objeto. Bajo tales condiciones, se introducen las siguientes reglas para tratar con solicitudes de acceso en conflicto.

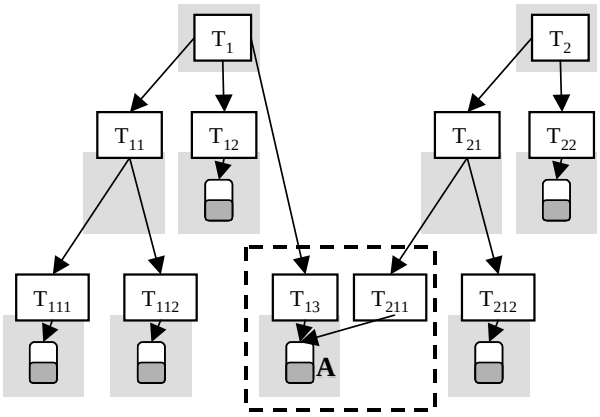


Figura 13. Ilustración de acceso concurrente: T_{13} se encuentra usando A cuando T_{211} trata de acceder al mismo objeto. El Administrador de Candados incrustado sigue algunas reglas sencillas pare resolver el conflicto.

Ejemplos

Para ilustrar el escenario arriba introducido, supongamos que en la Figura 13, el agente que monitoriza el nodo T_{13} ya ha bloqueado A para acceso exclusivo. Luego entonces, si una segunda transacción T_{211} llega con la intención de usar A, recibirá como una notificación de espera *wait* y T_{211} entrará a un estado de espera.

Mientras tanto, pueden ocurrir dos eventos: T_{13} libera el objeto A, el Administrador notifica este evento a T_{211}, por lo que puede continuar con la ejecución de las operaciones; segundo, a T_{21} se le vence el lapso de tiempo asignado en el cual, debe verificar si su hijo T_{211} está o no listo para confirmar y de esta forma reportar un resultado al nodo superior T_2. T_{211} no tiene otra opción que responder *No* y abortar. A continuación T_{21} debe decidir si es posible confirmar la sub-transacción. Esta decisión se toma de acuerdo a la clasificación propuesta como parte del mecanismo. De acuerdo al protocolo descrito en 3.3.2, T_2 eventualmente estará enterado del éxito o falla de la rama a cargo de T_{21}. Si el nodo comprometido (T_{211}) desencadenó un aborto global debido a su prioridad, entonces el agente que administra el nodo raíz T_2, reportará la situación al agente del usuario para saber si se desea programar un nuevo intento de ejecución de las tareas incompletas. En una alternativa de implementación posible, los nuevos intentos podrían realizarse de manera automática sin intervención del usuario hasta cierto número de veces, antes de cancelar definitivamente la transacción.

3.4 Capa de Base de Datos

Está compuesta primordialmente por los nodos hoja del árbol de la transacción. En este nivel se realiza físicamente la lectura o escritura sobre los objetos de datos. El primer intento para ejecutar el conjunto de operaciones SQL, se realiza cuando el nodo recibe los datos de la subtransacción. Las tareas que comprenden esta capa del monitor son:

a) Solicitar al administrador de candados, un candado para cada operación SQL de la lista,
b) Si fue posible obtener todos los candados, ejecutar las operaciones o
c) Si no fue posible obtener todos los candados, dar de alta un comportamiento *TimeOutManager* en caso de que el nodo esté autorizado para ello.

Durante el tiempo de espera, el agente puede ser notificado por el administrador de candados para que realice nuevos intento de ejecución de las operaciones. En este lapso, el agente pudo haber recibido la notificación CAN_COMMIT (posibilidad de compromiso) de parte del protocolo 2PC; en este caso, se informa al padre del resultado (YES / NO) obtenido. De lo contrario se regresa al estado de espera de la siguiente notificación de parte del administrador de candados o de su nodo padre.

El procesamiento de transacciones planas se considera un caso especial de la capa de base de datos, en el cual el nodo raíz es el único participante. Además de registrar los datos de la transacción y obtener un nivel de prioridad entre los cuatro posibles, el nodo se comporta como hoja en lo que se refiere a la ejecución de las instrucciones SQL sobre los datos. El resultado final se informa directamente al cliente que solicitó ejecutar la transacción.

La implementación del monitor para grupos de dispositivos móviles se apoya en el manejador de bases de datos Pointbase en su versión 4.6; está programado en Java, al igual que la plataforma de agentes seleccionada. Su tamaño, menor a 45Kb, le permite incrustarlo como parte del agente, así también las tablas de datos que se leen o modifican durante la ejecución de transacciones.

4 Experimentos y resultados

A continuación se presenta el entorno tecnológico para realización de los experimentos, los resultados obtenidos y su interpretación; resalta la tolerancia a fallos demostrada por el modelo de transacciones anidadas, los beneficios de distribuir la carga de trabajo en un mayor número de unidades así como las ventajas de ceder un mayor tiempo de oportunidad, para que las transacciones intenten completar operaciones que fallaron en su primer intento.

4.1 Descripción

4.1.1 Entorno tecnológico

El sistema multi-agente está desarrollado sobre la plataforma *Java Agent DEvelopment Framework* (JADE) 3.1, la cual habilita la realización y despliegue de agentes, conforme las especificaciones FIPA. Al estar basada en el lenguaje de programación Java, facilita la interacción de entidades distribuidas que residen en sistemas operativos heterogéneos. Asimismo, JADE se distribuye bajo la licencia LGPL, esto es, el código fuente se encuentra disponible, pero la plataforma puede emplearse en aplicaciones comerciales también. JADE incorporó el proyecto LEAP *Lightweight Extensible Agent Platform*, el cual es un subconjunto de las clases originales, adaptadas para su ejecución en dispositivos móviles.

Con LEAP es posible implementar agentes para la edición estándar de Java J2SE, así como también para los perfiles Personal Java (*Connected Devicce Configuration*) y Micro Information Device Profile MIDP. El segundo perfil consiste de clases Java que posibilitan la ejecución de ciertas operaciones en dispositivos con capacidad de cómputo restringida como los celulares y los asistentes digitales personales (PDA por sus siglas en inglés). En particular, MIDP define la especificación para almacenamiento, interfaz de usuario y comunicaciones entre dispositivos móviles. Los principales fabricantes de teléfonos celulares lo han adoptado en la implementación de las máquinas virtuales para sus dispositivos. Una aplicación desarrollada para el perfil MIDP se le conoce como midlet. Sun Microsystems ofrece al programador el J2ME Wireless Toolkit, el cual incluye un emulador de un teléfono genérico así como herramientas para construir y ejecutar midlets.

La ejecución de agentes LEAP en PDA's requiere de una máquina virtual acorde al sistema operativo del dispositivo anfitrión. En el caso de PalmOS, una opción es emplear MIDP4PALM [Midp05] y para WindowsCE se puede usar me4se [Me4se05].

Los experimentos del sistema multi-agente se realizaron con diferentes configuraciones que empleaban algunos de los siguientes dispositivos:

- teléfono Motorola C650,
- teléfono Siemens S40,
- teléfono SonyEricsson z600,
- teléfono Samsung x436,
- PDA PALM Tungsten,
- PDA Sony Cliè NRV70
- teléfono SunWireless Toolkit (emulado),
- PDA PalmOS Emulador (emulado).

Asimismo los agentes desarrollados con la edición estándar de Java, son ejecutables sobre computadoras portátiles conectadas en una red inalámbrica. Para fines de pruebas se emplearon computadoras personales Pentium IV conectadas a una red local.

En la Fig. 14 se ilustra un ejemplo de despliegue físico de dispositivos empleado en los experimentos del monitor. La transacción da inicio en un teléfono celular emulado (1) en representación de un dispositivo de usuario. La petición de ejecución recorre el camino a través de la red inalámbrica local, Internet y el servidor de comunicaciones de Telmex (los teléfonos involucrados se encontraban bajo el esquema de prepago de la compañía mencionada). Eventualmente, la solicitud alcanza al nodo que operará como raíz de la transacción, mismo que se encuentra alojado en un teléfono celular (2). Éste último, distribuye y coordina la ejecución de las operaciones entre nodos emulados (3.a) o reales (3.b).

El envío de paquetes de trabajo así como la notificación de resultados por parte de nodos hijo/hoja, implica transferencias adicionales de datos desde la computadora anfitrión hacia los dispositivos participantes y viceversa. En comparación, el procesamiento efectivo de las operaciones que conforman la transacción anidada, se lleva a cabo localmente sobre la base datos incrustada en cada dispositivo.

La participación de teléfonos celulares se da previa realización de llamadas telefónicas, las cuales abren canales de datos conmutados por circuitos (CSD por sus siglas en inglés). Una alternativa para la comunicación entre dispositivos móviles y redes alambradas, es mediante el sistema de radio general de paquetes (GPRS); éste habilita un enlace inmediato en tanto el dispositivo se encuentra dentro de la zona de cobertura. Sin embargo, al tiempo de realización de los experimentos, este servicio no se encontraba disponible en esquemas de prepago.

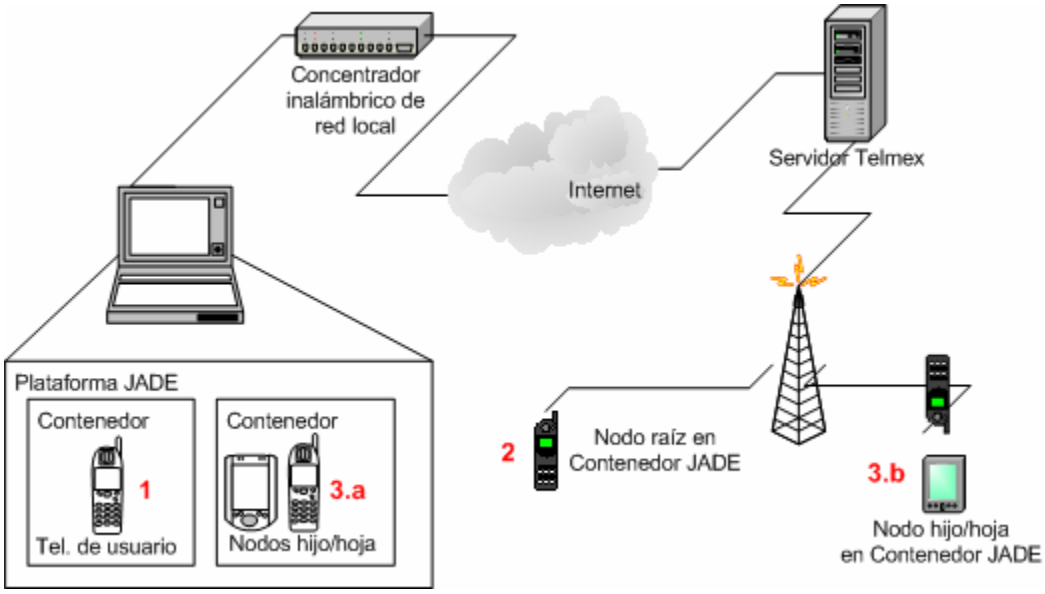


Figura 14. Ejemplo experimental de despliegue físico de dispositivos

4.1.2 Configuración de transacciones

Para evaluar el desempeño del monitor, se diseñaron cuatro configuraciones representativas de árboles de transacciones. Son árboles con diferente número de nodos intermedios y hojas, asimismo la profundidad es variable. Sin embargo, el monitor puede soportar un mayor número de niveles.

La transacción distribuida (Fig 15.a) representa la configuración del árbol donde no existen nodos intermedios y se prueba la expansión del árbol en anchura. A continuación se tiene la configuración Mixta (Fig. 15.b), que integra tanto hojas coordinadas directamente por el nodo raíz, como un subárbol que permite experimentar con un caso de anidamiento sencillo. El anidamiento completo se presenta en la configuración Árbol (Fig. 15.c), donde el monitoreo desde el nodo raíz hacia las hojas se realiza por intermedio de dos nodos padre. Finalmente, en la configuración Escalera (Fig. 15.d), con doble anidamiento y por tanto con mayor profundidad, se prueba la extensibilidad hacia abajo del grupo de trabajo.

De acuerdo al modelo de transacciones anidadas móviles, únicamente las hojas del árbol tienen acceso a los datos. Bajo esta consideración, se lanzaron cuatro transacciones concurrentes, una por cada tipo, que ejecutaron operaciones de escritura en las hojas. Una excepción se presenta con la configuración de Árbol, donde las hojas centrales ejecutaron operaciones de lectura. El total de operaciones que comprendía una transacción fue ocho, divididas en el número de hojas. Es decir, cada una de éstas últimas recibía dos instrucciones por transacción concurrente. El grupo de cuatro transacciones se probó en diferentes escenarios

que combinan agentes en dispositivos reales, emuladores y computadoras portátiles (PC).

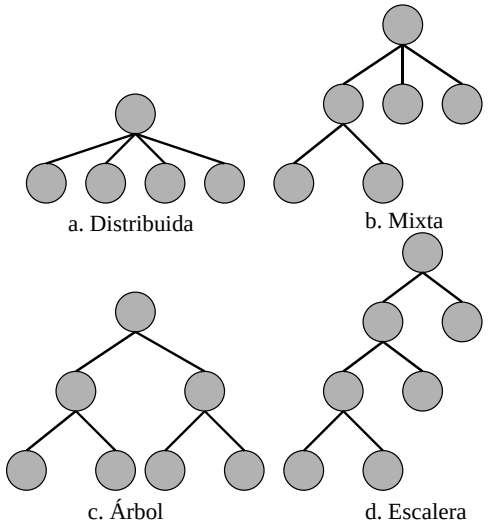


Figura 15. Configuraciones de transacciones para experimentos

En este punto, cabe recordar que cada dispositivo puede tomar el rol de raíz, participante u hoja. De acuerdo al diseño de los experimentos, un dispositivo puede tomar hasta un total de cuatro roles concurrentes en promedio.

Debido a las limitaciones en capacidad de cómputo y almacenamiento en los dispositivos, las pruebas de mayor concurrencia se realizaron sobre una computadora personal. De esta forma se evaluó la robustez y escalabilidad del modelo. Para ello se ejecutaron dieciséis transacciones concurrentes, cuatro de cada tipo que combinaban operaciones de lectura (L) y escritura (E) en las hojas como se detalla a continuación.

- Distribuida: LLLL, ELEL, LELE, EEEE.
- Mixta: LLLL, LELE, ELEL, EEEE.
- Árbol: LLLL, ELLL, LLLE, ELLE.
- Escalera: LEEL, ELEL, LELE, EEEE.

Finalmente, la base de datos incrustada en cada dispositivo, tiene una tabla con cuatro diferentes columnas (Figura 16). Cada una con un tipo de datos distinto: arreglo fijo de caracteres, un número entero, un valor tipo fecha y un número con decimales. Aunque la plataforma J2ME no soporta el punto flotante, Pointbase lo simula por medio de su tipo DECIMAL.

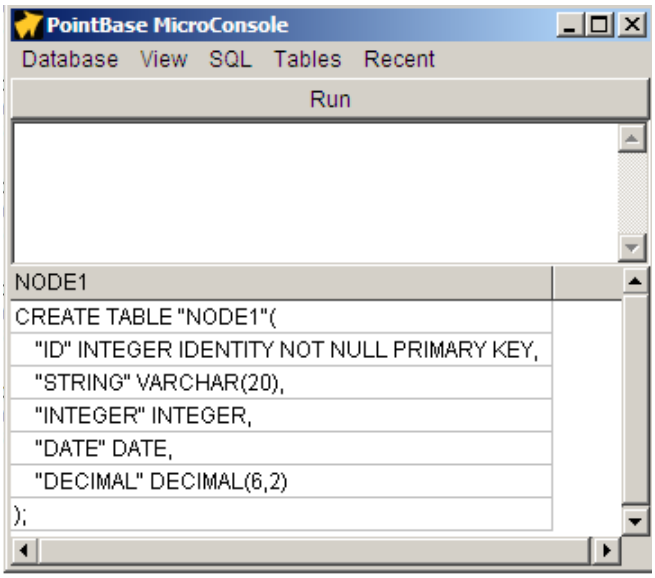


Figura 16. Descripción de las tablas incrustadas en el dispositivo *Node1*

Las figuras 17.a y 17.c ilustran un ejemplo del cambio de los datos en las tablas incrustadas en uno de los dispositivos emulados, antes y después de la ejecución de una transacción distribuida. La figura 17.b enlista las operaciones de escritura de datos que en particular, son enviadas al dispositivo *Node2* como parte de la transacción antes mencionada.

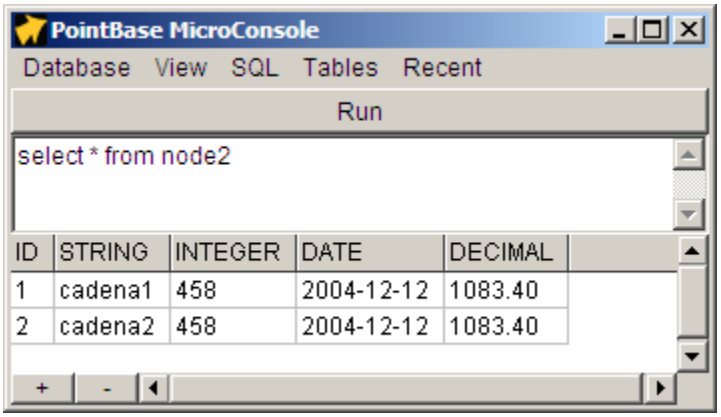


Figura 17.a. Ejemplo de estado de los datos en el dispositivo *Node2* antes de la ejecución de una transacción distribuida

```
…
a.add("UPDATE NODE2 SET INTEGER = INTEGER+1 WHERE ID=1");
a.add("UPDATE NODE2 SET INTEGER = INTEGER+1 WHERE ID=2");
…
```

Figura 17.b. Ejemplo de operaciones de escritura de datos ejecutadas en el dispositivo *Node2*

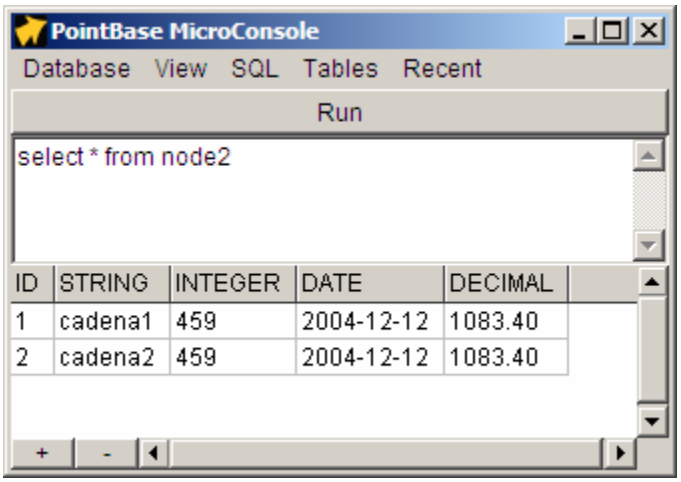


Figura 17.c. Ejemplo de estado de los datos en el dispositivo *Node2* después de la ejecución de una transacción distribuida

4.2 Resultados

En el caso de cuatro transacciones concurrentes, se realizaron pruebas sobre los siguientes escenarios:

- **a)** 1 agente en teléfono Motorola C650, un agente en PDA Sony Cliè conectado a la red Telcel GSM por medio del Siemens S40 y 5 agentes PC.
- **b)** 1 agente en teléfono Motorola C650 conectado a la red Telcel y 6 agentes PC,
- **c)** 1 agente en PDA Palm Tungsten conectado vía red inalámbrica y 6 agentes PC,
- **d)** 1 agente en PDA Palm Tungsten conectado a la red Telcel GSM por medio del SonyEricsson z600 y 6 agentes PC,
- **e)** 1 agente en emulador de teléfono y 6 agentes PC
- **f)** 1 agente en el emulador POSE y 6 agentes PC ,

La Tabla 2 muestra los tiempos aproximados de ejecución de las transacciones en distintos escenarios. Con excepción del escenario a), los resultados se expresan en segundos. Se muestra el gráfico comparativo para los escenarios b) a f) en la Figura 18.

Tabla 2. Tiempos de ejecución por configuración en cada escenario

	Escenario					
Configuración	a)	b)	c)	d)	e)	f)
Distribuida	4.6 min	5.2	22.1	76.8	3.0	7.4
Mixta	4.5 min	10.1	3.7	10.1	3.2	4.2
Árbol	5.1 min	37.6	5.8	10.2	3.4	3.3
Escalera	5.1 min	31.9	19.3	69.7	3.6	7.1

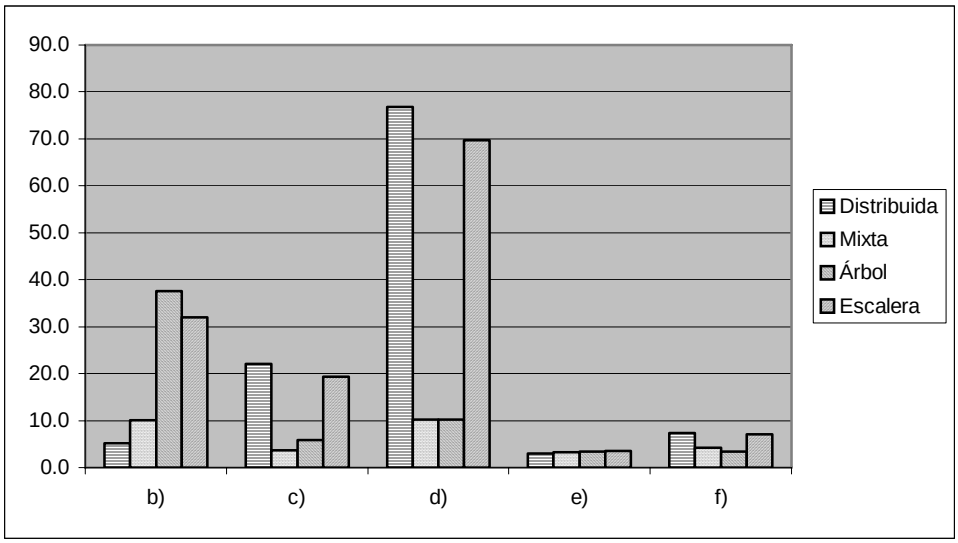


Figura 18. Tiempos de ejecución por configuración en cada escenario

La Tabla 3 muestra el porcentaje de transacciones exitosamente completadas según la configuración. Estos datos se ilustran en la Figura 19.

Tabla 3. Porcentaje de transacciones confirmadas por configuración en cada escenario

	Escenario			
Configuración	c)	d)	e)	f)
Distribuida	68.75%	0.00%	59.38%	37.50%
Mixta	81.25%	50.00%	71.88%	81.25%
Árbol	81.25%	75.00%	81.25%	68.75%
Escalera	87.50%	100.00%	68.75%	87.50%

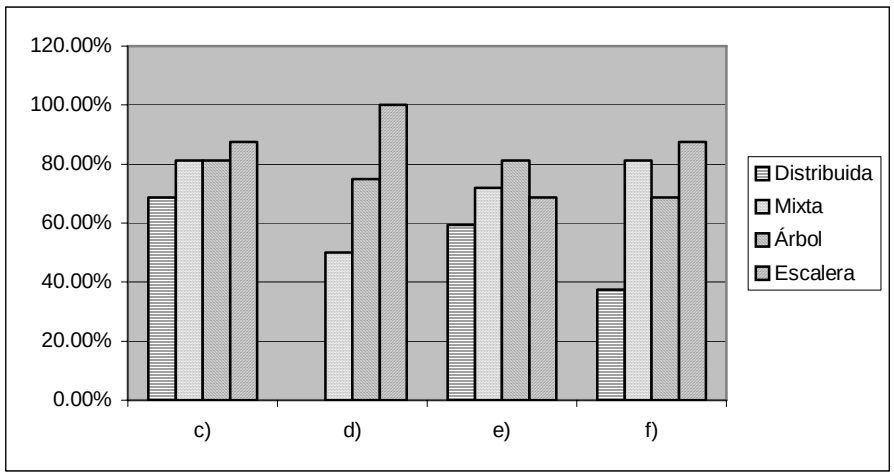


Figura 19. Porcentaje de transacciones confirmadas por configuración en cada escenario

La Tabla 4 muestra el porcentaje de transacciones exitosamente completadas de acuerdo al rol que jugó el agente para cada transacción concurrente. Los datos se ilustran en la Figura 20.

Tabla 4. Porcentaje de transacciones confirmadas por tipo de nodo en cada escenario

	Escenario			
Rol del agente	c)	d)	e)	f)
Raíz	79.69%	56.25%	70.31%	64.06%
Padre	64.06%	73.13%	61.33%	60.94%
Hoja	57.29%	49.31%	46.79%	50.52%

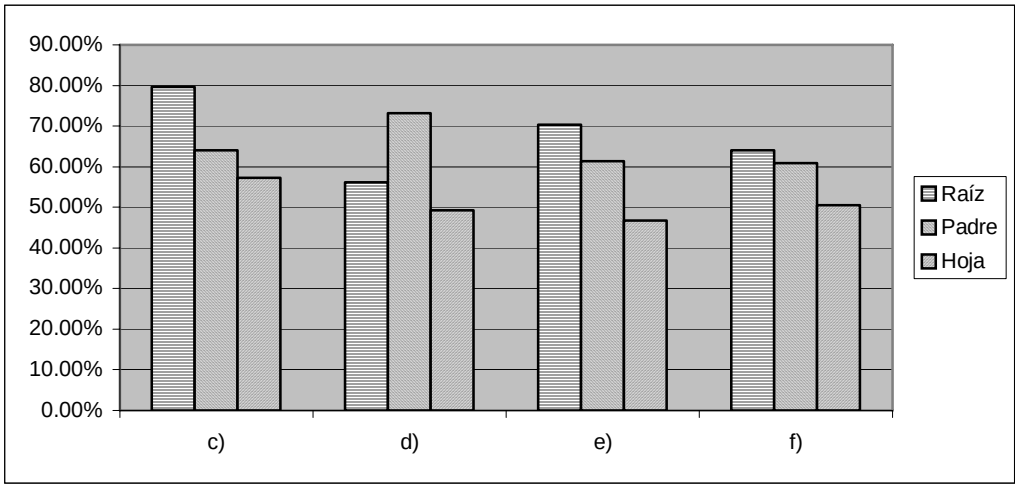


Figura 20. Porcentaje de transacciones confirmadas por tipo de nodo en cada escenario

La Tabla 5 muestra el porcentaje de transacciones exitosamente confirmadas en los experimentos. El comparativo gráfico está en la Figura 21.

Tabla 5. Porcentaje de transacciones confirmadas por clasificación del nodo en cada escenario

	Escenario			
Clasificación	c)	d)	e)	f)
Opcionales	74.90%	69.84%	88.55%	81.60%
Obligatorias (Débil)	57.00%	63.27%	36.41%	49.82%
Obligatorias (Fuerte)	83.02%	85.83%	83.79%	80.14%
Críticas	49.75%	67.36%	41.32%	49.75%

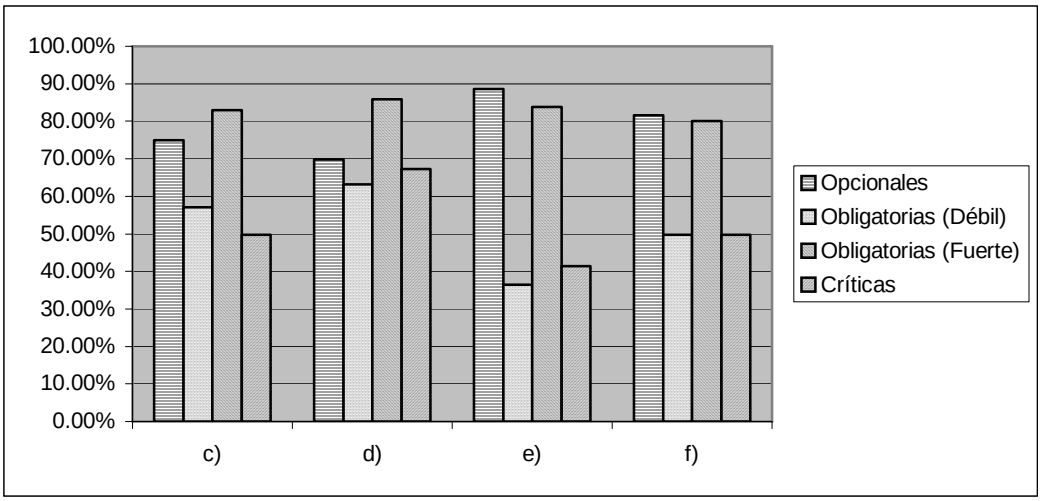


Figura 21. Porcentaje de transacciones confirmadas por clasificación del nodo en cada escenario

Se tienen también los resultados de aquellas transacciones a las que su padre les indicó una decisión distinta a la tomada localmente. El primer renglón menciona el porcentaje que estas transacciones representan contra el total administrado por los nodos localmente. Enseguida se tiene el porcentaje de transacciones a quienes su padre les indicó que confirmaran (COMMIT) a pesar que ellas, localmente, habían decidido abortar. Por último se tienen el porcentaje de transacciones a quienes su padre les indicó abortar a pesar que, localmente, habían decidido confirmar (COMMIT).

Tabla 6. Porcentaje de transacciones cuyo nodo monitor recibió una instrucción conclusiva distinta a la tomada localmente

	Escenario			
	c)	d)	e)	f)
% con respecto al total	19%	28%	24%	28%
Se le indicó COMMIT	25.34%	19.44%	18.70%	18.84%
Se le indicó ABORT	74.66%	80.56%	81.30%	81.16%

Finalmente, la tabla 7 y las figuras 22 a 25, muestran los resultados de experimentos donde se ejecutaron dieciséis transacciones concurrentes.

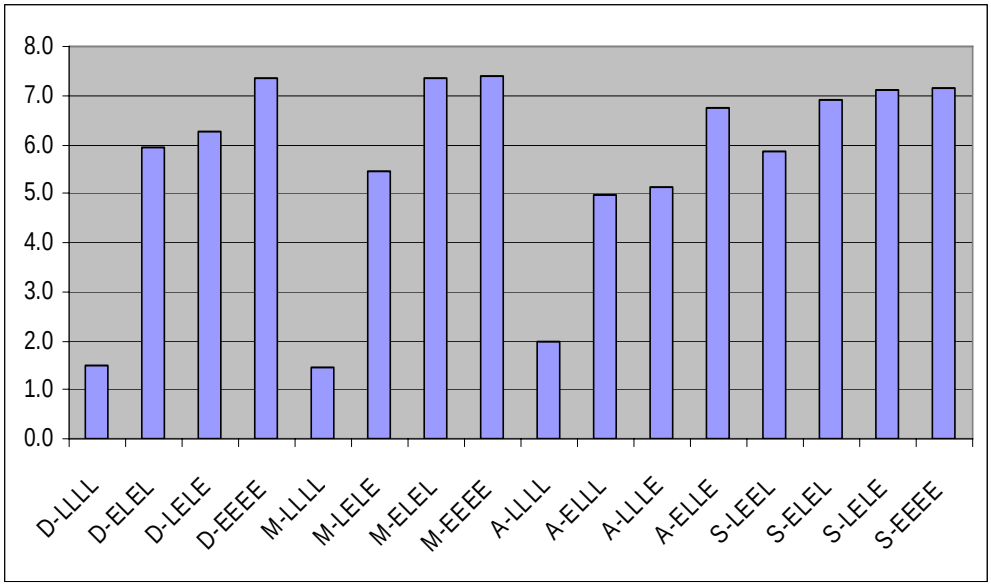


Figura 22. Tiempos de ejecución promedio de las 16 transacciones concurrentes

Tabla 7. Porcentajes de confirmación en los experimentos con 16 transacciones concurrentes

Por clasificación del nodo	Opcional	Obligatorio Débil	Obligatorio Fuerte	Crítico
	75.06%	51.80%	72.49%	51.04%

Por configuración de transacción	Distribuidas	Mixtas	Árboles	Escaleras
	44.32%	55.68%	85.80%	55.68%

Por rol del nodo	Raíz	Padre	Hoja
	60.37%	61.64%	42.32%

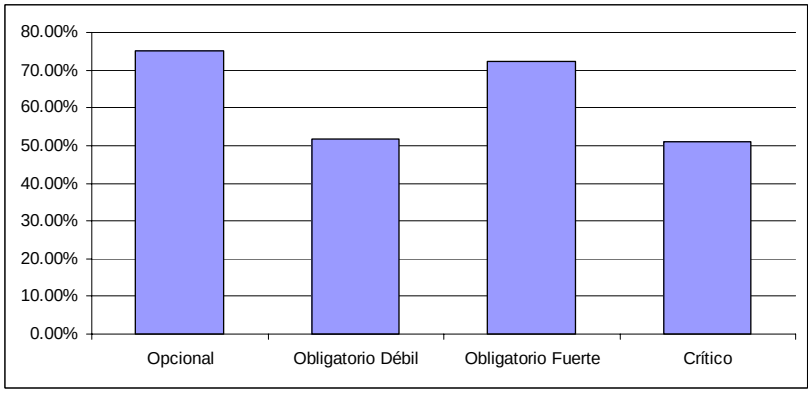


Figura 23. Porcentajes de confirmación por *clasificación del nodo* con 16 transacciones concurrentes

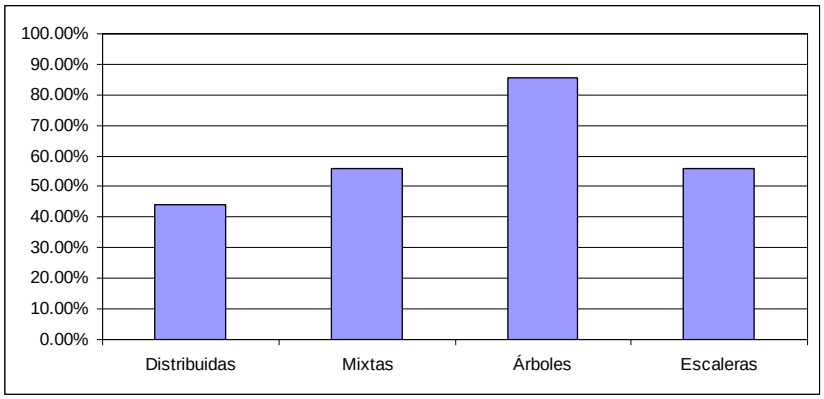


Figura 24. Porcentajes de confirmación por *configuraciones* de 16 transacciones concurrentes

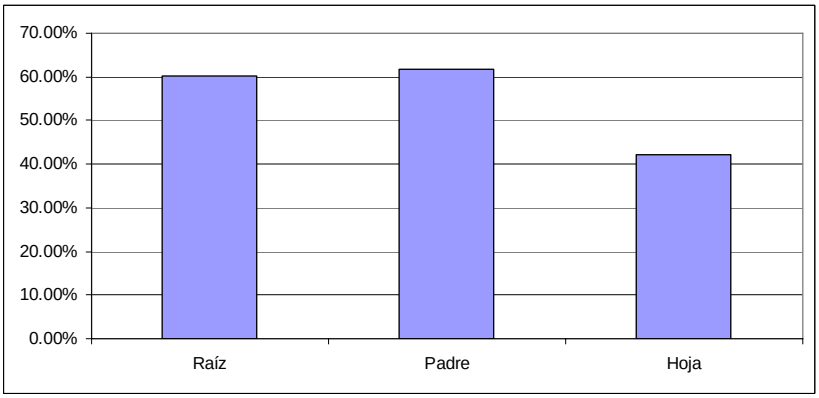


Figura 25. Porcentajes de confirmación por *rol del nodo* en 16 transacciones concurrentes

4.3 Análisis

En los resultados obtenidos, se aprecia que el experimento más tardado es donde se combina un teléfono celular, una PDA conectada a Internet por medio de la red GSM y 5 agentes residentes en computadora personal. Mientras que en otros experimentos, una transacción se tomó entre 3 a 76.8 segundos para concluir, en el caso antes mencionado, se requirieron más de 4 minutos y medio para completar las configuraciones Mixta y Distribuida, y más de 5 minutos para completar las configuraciones Árbol y Escalera.

Enseguida se tienen las combinaciones b) y d), las cuales comparten el hecho de tener un dispositivo en el grupo conectado a la red GSM. Sin embargo, se aprecia que el desempeño del teléfono celular mejora los tiempos obtenidos en los experimentos con una PDA. La misma situación se presenta en los experimentos con emuladores donde, las transacciones del grupo con un teléfono (e),

requirieron, en promedio, menos tiempo para concluir que las transacciones del grupo con una PDA (f).

Una de las explicaciones para estos resultados, se encuentra en las limitaciones de la máquina virtual MIDP4PALM. Ésta opera sobre el sistema operativo POSE de la PDA, por lo que representa una capa adicional para la ejecución de los midlets. En contraste, el teléfono físico incorpora directamente en *hardware* la máquina virtual para J2ME y el teléfono emulado representa por sí mismo una máquina virtual para el despliegue de midlets. Adicionalmente, la PDA es dependiente de un segundo teléfono celular en funciones de módem para conexión a la red GSM. En cambio, los midlets en ejecución directa sobre el teléfono celular, emplean el hardware (real o emulado) como su medio de acceso a la red de comunicaciones.

Otro resultado ilustra la disparidad de los tiempos de ejecución en transacciones que pertenecen al mismo grupo. Por ejemplo, en el caso del experimento b), mientras que la transacción distribuida tomó en promedio 5.2 segundos para completar, y la mixta requirió 10.1 segundos, tanto el árbol como la escalera tardaron más de 30 segundos. Una situación similar se presenta con los experimentos c) y d), en donde las configuraciones distribuida y escalera, exceden marcadamente el tiempo de ejecución con respecto a las configuraciones mixta y árbol. En general, las transacciones que tardaron más en completarse, son aquellas donde alguno de sus nodos participantes fue monitorizado por un dispositivo (real o emulado). En contraste, los tiempos más homogéneos se presentaron en el experimento e), con 1 emulador de teléfono y 6 agentes PC.

Las transacciones completadas exitosamente se consideran aquellas que ejecutaron la operación COMMIT para confirmar las operaciones. En los resultados, la configuración de Escalera presenta el mejor desempeño en general. La excepción se encuentra en el escenario donde participó un emulador de teléfono celular. Aquí, el mejor porcentaje de transacciones exitosamente completadas corresponde a la configuración de Árbol quedando la Escalera en 2º lugar. El peor desempeño en todos los experimentos lo tiene la configuración Distribuida. Esto se debe a que todas las hojas se encuentran en el mismo nivel. Si alguna de ellas aborta, estando clasificada como nodo Crítico u Obligatorio Fuerte, toda la transacción coordinada por el nodo raíz aborta en consecuencia.

En general, la configuración de Árbol obtuvo los siguientes mejores desempeños por detrás de la Escalera. En los experimentos desarrollados, las cuatro hojas de una transacción tipo Árbol son monitorizadas por tres agentes. En contraste, las cuatro hojas de una transacción Mixta son monitorizadas por cuatro agentes. Sin embargo, en el Árbol se presentan dos nodos intermedios. Con respecto a la configuración Mixta, esto significa una mayor división en el número de agentes

que monitorizan hojas. No obstante, el Árbol no cuenta con la profundidad característica de la configuración Escalera.

Con estos resultados se puede afirmar que:

- la distribución de piezas de trabajo en ramas con mayor profundidad, impacta positivamente el desempeño global de las transacciones;
- el resultado final de la transacción no queda bajo responsabilidad de un solo agente (como sucede en la configuración Distribuida); por el contrario,
- la división de la transacción original en varias subtransacciones, permite tomar en cuenta decisiones locales de los agentes que monitorizan nodos intermedios;
- bajo las reglas definidas para el control de concurrencia, estos agentes cuentan con autonomía para reportar su propia decisión de acuerdo a lo informado por los nodos bajo su supervisión y,
- con la participación de un mayor número de agentes monitorizando nodos de la transacción, se puede mejorar el porcentaje global de transacciones confirmadas.

En general los resultados de los experimentos por tipo de nodo (raíz, padre, hoja), muestran que a pesar del porcentaje de transacciones confirmadas por las hojas (entre el 46.79% y el 57.29%), los nodos intermedios presentan mejores tasas de éxito e incluso se ven superadas por los porcentajes de los nodos raíz (hasta un 76.69%).

Estos resultados son un reflejo de cómo influye la tabla de clasificaciones (ver 3.3.4) en el comportamiento global de la transacción. De acuerdo a las reglas implementadas, el hecho que un nodo hijo falle, no es condicionante para abortar la rama de la transacción. Por ello, las raíces de la transacción alcanzan los mejores porcentajes de confirmación a pesar que en los nodos intermedios, apenas en cada 3 de cada 5 transacciones, se envió una respuesta de tipo SI a la petición *CanCommit*. En el nivel de hojas, esto sólo ocurría en 1 de cada 2 transacciones procesadas.

Los porcentajes de éxito en los resultados por clasificación del nodo, muestran un desempeño homogéneo entre los distintos escenarios. Al respecto, tanto los nodos Opcionales como los Obligatorios (Fuerte) alcanzaron mayores niveles de éxito en comparación a los nodos de tipo Obligatorio (Débil) y Críticos. Como se mencionó en la Sección 3.3.3, la ejecución de transacciones en estos dos últimos casos, está condicionada a la disponibilidad inmediata de los recursos. En contraste, los nodos Opcionales y Obligatorios (Fuerte) se ven favorecidos por la posibilidad de contar con un tiempo extra para realizar nuevos intentos de ejecución.

Finalmente se tienen los resultados de aquellas transacciones a quienes la decisión final de su nodo raíz, fue diferente a la reportada originalmente durante el protocolo de compromiso atómico. En promedio, 1 de cada 4 transacciones procesadas por el agente concluyó con un resultado distinto al reportado por dicho agente. De este porcentaje, aproximadamente, a 1 de cada 5 transacciones se le solicitó confirmar a pesar que originalmente iban a abortar la rama. Al resto se les solicitó abortar a pesar que su intención era confirmar la rama de la transacción.

En el escenario c) los porcentajes varían con respecto a los demás experimentos. Allí, 1 de cada 3 transacciones recibió una decisión final del nodo raíz distinta a la decisión local. De este porcentaje, en promedio 1 de cada 4 transacciones confirmó mientras que el resto abortó.

Estos resultados se apegan al comportamiento esperado de las transacciones anidadas; esto es, la imposibilidad de ejecutar completamente todas las operaciones originales, ya no es condicionante para abortar la transacción global. El monitor desarrollado, permite incluso enviar el conjunto de operaciones incompletas al cliente para que decida si desea programar una nueva transacción; por medio de esta última podría completarse el conjunto original. Los porcentajes reflejan asimismo, el balance de autonomía de los agentes con respecto a la decisión final que toma el nodo raíz de la transacción. Los resultados en estos experimentos abren espacio para desarrollos posteriores con el esquema de transacciones anidadas abiertas. Allí los nodos intermedios y hojas pueden confirmar o abortar definitivamente su rama de la transacción sin esperar la notificación final de la raíz.

En los **experimentos con 16 nodos,** los tiempos de ejecución promedio (desde la perspectiva de rapidez) muestran que en transacciones de sólo lectura, el mejor desempeño lo tiene la configuración Mixta; en comparación, para transacciones que requieren escribir sobre los datos, la configuración de Escalera resalta ligeramente sobre los otros modelos. Sin embargo, es importante remarcar que el mayor porcentaje de transacciones confirmadas se encuentra en las configuraciones de tipo Árbol. En este punto, es posible afirmar que el incremento en la concurrencia, puede verse beneficiado por la distribución de piezas de trabajo en ramas equilibradas, no necesariamente con mayor profundidad – como sucedió en los experimentos con cuatro transacciones.

Asimismo, se muestran porcentajes muy similares de éxito según el rol del agente, cuando éste operó como raíz o padre (nodo intermedio). Esto contrasta con los primeros experimentos donde, a pesar de las fallas en nodos hijos, las raíces lograban confirmar un mayor número de transacciones. En este caso, los nodos raíz muestran incluso un menor porcentaje de transacciones confirmadas. En otras palabras, las transacciones globales abortan a pesar de que en nodos intermedios se

tomó la decisión local de confirmar. Aproximadamente, en 1 de cada 3 casos, los nodos recibieron una decisión distinta de la reportada durante el protocolo de compromiso atómico. De esta fracción, en 9 de cada 10 casos, se le solicitó al agente abortar la transacción aunque éste había decidido confirmar.

5 Discusión y perspectiva

En el presente trabajo se experimentó con la integración de diferentes tecnologías; por una parte los sistemas multi-agente, cuyo potencial a la fecha, no ha sido del todo explotado exitosamente desde el punto de vista comercial. Por otra parte, el cómputo móvil, que actualmente se apoya en una infraestructura respetable de telecomunicaciones y cuenta con una oferta considerable de dispositivos. No obstante, aún es difícil asegurar que se tiene una integración transparente en las actividades diarias del consumidor; al menos no en la visión que prometen el cómputo ubicuo y el penetrante. Enseguida, se tiene el modelo de transacciones anidadas, cuyas ventajas mostradas (en la teoría), difícilmente compiten con la popularidad de los modelos transaccionales (no anidados) implementados en los manejadores de bases de datos comerciales.

Flexibilidad del modelo y limitaciones tecnológicas

La revisión de literatura al respecto permite afirmar que el presente trabajo no es único ni aislado. Sin embargo, de forma coincidente, en todos estos desarrollos del ámbito académico, aún no se vislumbra claramente, su implementación en aplicaciones para el consumidor comercial. En esta tesis la base teórica del monitor de transacciones, hace uso de una Lógica de Interacción y del modelo de Transacciones Anidadas Móviles. Las complicaciones notorias en la operación del sistema son de naturaleza práctica. En primer lugar, no fue posible instalar la aplicación en los dispositivos de prueba por la vía inalámbrica –que es el método usado para descargar contenido a teléfonos celulares. Con un tamaño que se aproxima a los 700 KB, el sistema es muy pesado en el ámbito de las aplicaciones móviles. La inestabilidad del servicio tampoco garantizó el procesamiento exitoso de todas las transacciones. La versión empleada JADE-LEAP no soporta de manera transparente desconexiones totales de los agentes con respecto al contenedor central; en contraste, vale la pena resaltar su protocolo de comunicación [Caire02b], mismo que soporta desconexiones temporales por unos cuantos segundos. Para flexibilizar la administración del tiempo de desconexión es necesario modificar a la plataforma.

Entre las implicaciones prácticas en la operación del sistema resalta el alto costo de operación. El precio del minuto de conexión de datos en la red GSM Telcel (la más popular y extendida en México) se encuentra actualmente en $1.50 más impuestos. Una transacción de 30 segundos consumiría alrededor de $50, sólo en tiempo aire. La máquina virtual MIDP4PALM, necesaria para ejecutar el monitor sobre PDAs, está limitada en sus capacidades. Con apenas 64 KB de memoria disponible para las aplicaciones, no fue posible que los agentes soportaran más de dos roles en

forma simultánea. En contraste, los agentes en teléfonos (real y emulado) no tuvieron problemas hasta con 5 roles simultáneos. Una carga de trabajo similar se esperaría en el despliegue de la aplicación en el mercado. Las características de MIDP4PALM no han cambiado desde su liberación. Una razón de peso para ello, es la importante evolución en la capacidad de las máquinas virtuales Java incrustadas en los teléfonos celulares. Por ejemplo, la unidad Motorola C650 cuenta con 1 MB de memoria para aplicaciones. Las mejoras que se introducen continuamente en teléfonos celulares, han relegado la opción de emplear PDAs para ejecutar midlets.

Puntos prometedores

La infraestructura basada en agentes se ve favorecida por las nociones de autonomía y habilidades sociales en la interacción entre nodos. Asimismo, la naturaleza cooperativa de los agentes permite distribuir una tarea en asignaciones más pequeñas; esta es una característica esperada en ambientes de cómputo móvil [Mazer95]. La ejecución y coordinación son atendidas por todo el grupo y no sólo por una entidad central. Dado que en los experimentos se hizo evidente las limitantes de las capacidades de los dispositivos móviles, se comprueba la distribución de tareas como una alternativa útil para superar tales restricciones.

A partir de los resultados obtenidos, se identifican dos aspectos relevantes por desarrollar.

1. Explotación del trabajo asíncrono. Esto es, la posibilidad de que los nodos continúen el procesamiento de transacciones fuera de línea y, una vez reestablecida la comunicación, el grupo resuelva la transacción de acuerdo al nuevo estado alcanzado.
2. Experimentar con mayor autonomía de los nodos. Los resultados confirmaron que el éxito global de una transacción anidada no está del todo condicionado por fallas en las ramas. Asimismo, se encontró que en repetidas ocasiones, los agentes estaban dispuestos a confirmar localmente las subtransacciones, aunque al final recibieron una notificación de aborto por parte del nodo raíz. Por ello vale la pena experimentar hasta dónde es posible otorgar independencia a los nodos, bajo políticas ó normas que garanticen eficiencia grupal.

La conclusión relevante es: el progreso en el campo del cómputo móvil es base para la construcción de ambientes de cómputo ubicuo y penetrante. Las oportunidades se presentan más allá de la perspectiva de ingeniería, desarrollo e investigación. El negocio de las aplicaciones móviles es una realidad que demanda constante aprovisionamiento de tecnologías de soporte. Las cifras hablan por sí solas: el número de usuarios de teléfonos celulares creció exponencialmente de 11 millones en 1990 a 400 millones en 1999 [Economist99] y se encuentra en un estimado actual de 1000 millones a escala mundial.

6 Conclusiones

En la presente tesis se implementó un monitor de procesamiento transacciones anidadas sobre un grupo de dispositivos móviles. Las tres capas del monitor (Presentación, Flujo de Trabajo y Base de Datos), están apoyadas en un Sistema Multiagente. Los agentes están habilitados con un conjunto de comportamientos que se activan de acuerdo al rol que cada uno juega dentro de la transacción.

El fundamento teórico es la Lógica de Interacción (LoI), modelo formal que hace hincapié sobre las acciones en secuencia o en paralelo, concurrentes y sincronizadas, que los agentes llevan a cabo para alcanzar sus metas. La plataforma de desarrollo es Java Agent DEvelopment Framework en su versión adaptada para dispositivos móviles: Lightweight Extensible Agent Platform.

El protocolo de compromiso atómico recolecta los resultados de la transacción, de tal forma que el nodo raíz toma la decisión final sobre el resultado de la transacción. El administrador de candados administra el control de concurrencia en el acceso a datos compartidos. Este control también se apoya en una tabla de clasificación de acciones, así como las reglas que aplican para la solución de conflictos. El método aplicado ayuda en la decisión sobre si es posible confirmar una transacción a pesar de la falla en algunas operaciones, y si es necesario calendarizar nuevamente a estas últimas para una nueva ejecución.

Actualmente, las pruebas de Bases de Datos se llevan a cabo con el manejador reducido Pointbase que junto con las tablas van incrustados en el dispositivo móvil. Los experimentos incluyeron plataformas heterogéneas de hardware y software, desde emuladores hasta dispositivos reales conectados por una red de área local inalámbrica y por la red de servicios Telcel GSM.

Por las configuraciones empleadas para la experimentación, se confirma que la división del trabajo en unidades anidadas mejora el desempeño de la aplicación en general. Los resultados también confirman la mayor tolerancia a fallos por parte del modelo de transacciones anidadas y la viabilidad de implementar agentes de propósito específico sobre dispositivos móviles. Sin embargo, también se evidencian las dificultades técnicas por limitaciones de los dispositivos e inestabilidad de los servicios inalámbricos. Finalmente, se dejan abiertos dos campos para nuevos desarrollos: la explotación del trabajo asíncrono y la experimentación con mayor autonomía (en toma de decisiones) por parte de los nodos de la transacción

Publicaciones

Martinez, J., Alvarado, M.: Mobile Nested Transaction Monitor based on MultiAgent Systems: Workflow Layer and JADE ontologies. En L. Sheremetov, M. Alvarado (eds.) Proc. of Workshop on Intelligent Computing. Asociado a MICAI 2004. IMP-SMIA. Abril (2004) 351-365

Martinez, J., Alvarado, M.: Mobile Nested Transactions Monitor based on Multi-Agent Systems: Workflow Layer. En A. Gelbukh, G. Sidorov, W. A. Olán Cristóbal, J. A. Vera Félix (eds.) Recientes Avances en la Ciencia de la Computación en México. IPN-CIC. Mayo (2004) 38-47

Martínez-Muñoz, J., Gama-Moreno, L. A.: Administrador de Candados para Transacciones Anidadas sobre Dispositivos Móviles. Memorias del 2o. Taller de Cómputo Móvil. Asociado al Encuentro Nacional de Computación 2004. Sept. (2004)

Martinez, J., Alvarado, M.: Transaction Processing Monitor based on Autonomous Agents for Concurrency Control of Mobile Nested Transactions. Proc. 5th Iberoamerican Workshop on Multi-Agent Systems. Asociado a IBERAMIA 2004. Nov. (2004)

Martinez, J., Alvarado, M.: Concurrency Control for Nested Transactions based on Autonomous Agents. Inteligencia Artificial, Revista Iberoamericana de I. A. No. 25. Vol. 9. AEPIA, (2005) 29-38

Referencias

[Alvarado03] Alvarado, M., Sheremetov, L.: Modal Structure for Agents Interaction Based on Concurrent Actions. In V. Marîk, J. Müller, M. Pchouek (eds.): Multi-Agent Systems and Applications III: 3rd International Central and Eastern European Conference on Multi-Agent Systems. Lecture Notes in Computer Science, Vol. 2691. Springer-Verlag, Berlin Heidelberg New York (2003) 29-39

[Alvarado02] Alvarado, M., Sheremetov, L., German, E., Alva, E.: Logic of Interaction for Multiagent Systems. In: C.A. Coello Coello, A. de Albornoz, L.E. Sucar, O.C. Battistutti (eds.): MICAI 2002: Advances in Artificial Intelligence: Second Mexican International Conference on Artificial Intelligence. Lecture Notes in Computer Science, Vol. 2313. Springer-Verlag, Berlin Heidelberg New York (2002) 378-396

[Bacon02] Bacon, J.: Toward pervasive computing. Pervasive Computing, IEEE Vol. 1, Issue 2, Abr.-Jun. (2002) p. 84

[Bellifemine99] Bellifemine, F., Poggi, A., Rimassi, G.: JADE: A FIPA-Compliant agent framework, Proc. Practical Applications of Intelligent Agents and Multi-Agents, Abril (1999) 97-108

[Bernstein97] Bernstein, P. A., Newcomer, E.: Principles of Transaction Processing. Morgan Kaufmann Publishers Inc. (1997)

[Borriello02] Borriello, G.: Key challenges in communication for ubiquitous computing. Communications Magazine, IEEE. Vol. 40, Issue 5, Part Anniversary. Mayo (2002), 16-18

[Caire02] Caire, G.: JADE Tutorial: Application-defined content languages and ontologies. TILab S.p.A. (2002)

[Caire02b] Caire, G., Lhuillier, N., Rimassa, G.: A communication protocol for agents on handheld devices. Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices, Bologna (2002)

[Coulouris02] Coulouris G., Dollimore J. , Kindberg T.: Distributed Systems. Addison Wesley (2002)

[Economist99] The world in your pocket. The Economist, (1999) en http://www.economist.com/surveys/displayStory.cfm?story_id=246137

[vanEijk02] van Eijk, R.J., Ebben, P.W.G., Bargh, M.S.: Implementation of a scheduler agent system for traveling users. In proc. of Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices. Bologna (2002)

[Engelman00] Engelmann, R. "The origins of radio." IEEE Potentials, Oct/Nov (2000)

[Finin02] Finin, T., Joshi, A., Kagal, L., Ratsimor, O., Avancha, S., Korolev, V., Chen, H., Perich, F., Cost., S.: Intelligent Agents for Mobile and Embedded Devices. International Journal of Cooperative Information Systems (2002)

[Fischer02] Fischer, K., Hutter, D., Klush, M., Stephan, W.: Towards secure mobile multiagent based electronic marketplace systems. Electronic Notes in Theoretical Computer Science. Vol. 63. Elsevier Science (2002)

[Forman94] Forman, G. H.; Zahorjan, J.: The challenges of mobile computing. Computer. Volume 27, Issue 4, Abr. (1994) 38 – 47

[Freeland01] Freeland, M., Mat-Amin, H., Teangtrong, K., Wannalertsri, W., Wattanakasemsakul, U.: Pervasive computing: business opportunity and challenges. Management of Engineering and Technology, 2001. PICMET '01. Portland International Conference on. Vol. 1, 29 Jul.-2 Ago. (2001) p. 85

[Gama02] Gama, L. A., Alvarado, M.: Transacciones para Cómputo Móvil: presente y perspectiva futura. Revista Digital Universitaria, Vol. 3. No. 4. http://www.revista.unam.mx (2002)

[Gama03] Gama, L. A., Alvarado, M.: Concurrency control for Read-Only in Mobile Nested Transactions. In Braunschweig, B., Alvarado, M., Bañares-Alcantara, R., Sheremetov, L. (eds.): Proc. of Intelligent Computing in the Petroleum Industry ICPI'03, associated with IJCAI'03. (2003), 89-93 Disponible en: http://www.imp.mx/icpi/docs/ICPI03%20Workshop.pdf

[Gama04] L. A. Gama, M. Alvarado. Mobile Nested Transactions for Nomadic Teams. In: Alvarado, M., Sheremetov, L., Cantu, F.: Special Issue on Intelligent Computing for Petroleum Industry. Expert Systems with Applications 26-4. Elsevier Science. (2004) 105-113

[Gray93] J. Gray, A. Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, Inc. (1993)

[Gray96] R. Gray, D. Kotz, S. Nog, D. Rus, G. Cybenko. Mobile agents for mobile computing. Technical Report PCS-TR96-285, Dept. of Computer Science, Dartmouth College. (1996) Disponible en: ftp://ftp.cs.dartmouth.edu/TR/TR96-285.pdf

[Härder93] T. Härder, K. Rothermel. "Concurrency Control Issues in Nested Transactions". The VLDB Journal - The International Journal on Very Large Data Bases, Vol. 2, Issue 1, Springer-Verlag New York, Inc. (1993) 39-74

[Hassanein00] Hassanein, H. S., El-Sharkawi, M. E.: Performance Modeling of Nested Transactions in Database Systems. Proc. of the 2000 conference of the Centre for Advanced Studies on Collaborative research, IBM Press (2000) p. 4

[Kindberg02] Kindberg, T., Fox, A.: System software for ubiquitous computing. Pervasive Computing, IEEE. Vol. 1, Issue 1, Ene.-Mar. (2002) 70 - 81

[Kempster99] Kempster, T., Stirling, C., Thanisch, P.: Diluting ACID. ACM Sigmod Record, 28-4. pp. 17-23. ACM Press (1999)

[Kung81] Kung, H. T., Robinson, J. T.: Optimistic methods for concurrency control. ACM Transactions on Database Systems. Vol. 6, No. 2. (1981) 213-239

[Lo92] Lo, M. L., Ravishankar, C. V.: "A Concurrency Control Protocol for Nested Transactions". IBM Centre for Advanced Studies Conference. Proc. of the 1992 conference of the Centre for Advanced Studies on Collaborative research, Vol.2, IBM Press. (1992) 67-80

[Loke02] Loke, S. W.: Supporting Intelligent BDI Agents on Resource-Limited Mobile Devices - Issues and Challenges from a Mobile Database Perspective. In proc. of Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices. Bologna (2002)

[Mazer95] Mazer, M. S., Kermani, P., Chang, H.: Issues in Mobile Computing Systems, Guest Editors' Note. Personal Communications, Vol. 2, Issue 6, Dec. (1995) p. 12

[Me4se05] Running me4se on PocketPC (WindowsCE) en http://kobjects.sourceforge.net/me4se/pocketpc/

[Midp05] MIDP for PALM OS en http://java.sun.com/products/midp4palm

[Mitchell82] Mitchell, J. G., Dion J.: A Comparison of two network-based file servers. Comms. ACM, Vol. 25, No. 4. (1982) 233-245

[Moss85] Moss, J. E. B. Nested Transactions: An Approach to Reliable Distributed Computing. MIT Press, Cambridge, MA. (1985)

[Nagi01] Nagi, K.: Transactional Agents: Towards a Robust Multi-Agent System (ed). Lecture Notes in Computer Science Vol. 2249. Springer-Verlag, Berlin Heidelberg New York (2001)

[Niemela04] Niemela, E., Vaskivuo, T.: Agile middleware of pervasive computing environments. Pervasive Computing and Communications Workshops, 2004. Proc. of the Second IEEE Annual Conference on. 14-17 Mar. (2004) 192-197

[Orfali95] Orfali, R., Harkey, D., Edwards, J.: "Intergalactic Client/Server Computing". BYTE Special Report (1995) en
http://www.byte.com/art/9504/sec11/art1.htm

[Özsu99] Özsu, M. T., Valduriez, P.: Principles of Distributed Database Systems. 2nd Ed., Prentice-Hall, Inc. (1999), 381-401

[Park04] Park, J. T., Yamada, S., Brunner, M., Wade, V.: Panel two: Ubiquitous computing and communications: challenges in the management of ubiquitous computing and communication. Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP. Vol. 1, 19-23 Abril (2004), p. 914

[Pitoura95] Pitoura, E., Bhargava, B.: A Framework for Providing Consistent and Recoverable Agent-Based Access to Heterogeneous Mobile Databases. ACM SIGMOD Record, Vol. 24, Issue 3, ACM Press, Sept. (1995), 44-49

[Pointbase04] Pointbase Micro Developer's Guide at
http://www.pointbase.com/support/docs/pbmicro.pdf

[Racherla96] Racherla, G., Das, A.: Mobile computing-a promising future that still requires much work. Potentials, IEEE. Vol. 15, Issue 4, Oct-Nov (1996), 13 - 15

[Satyanarayanan93] Satyanarayanan, M.: Mobile computing. Computer. Vol. 26, Issue 9. Sept. (1993), 81 - 82

[Seifert03] Seifert, A., Scoll, M. H.. Processing read-only transactions in hybrid data delivery environments with consistency and concurrency guarantees. Mobile Networks and Applications, 8-4. Kluwer Academic Publishers. (2003), 327-342

[Stallings01] Stallings, W. Wireless Communications and Networks. Prentice Hall. New Jersey. (2001)

[Straßer98] Straßer, M., Baumann, J., Hohl, F., Schwehm, M.: ATOMAS: A Transaction-oriented Open Multi Agent-System. Final Report. Institute for Parallel and Distributed High Performance Systems, University of Stuttgart. (1998)

[Vogler98] Vogler, H.; Buchmann, A.: Using multiple mobile agents for distributed transactions. In proc. of 3rd IFCIS International Conference on Cooperative Information Systems. New York. (1998) 114-121

[Weiser91] Weiser, M. "The Computer for the Twenty-First Century." Scientific American. Vol. 265, Núm. 3. Sept. (1991) 94-104

[Weiser93] Weiser, M.: Hot topics-ubiquitous computing. Computer. Vol. 26, Issue 10, Oct. (1993), 71 - 72

[Wooldridge01] Wooldridge, M.: An Introduction to Multi-Agent Systems. John Wiley & Sons. England (2001)

[Zaslavsky98] Zaslavsky, A., Tahir Z: Mobile Computing: Overview and Current Status. Australian Computer Journal. Vol. 30. No. 2 (1998)