BESA/ME: Plataforma para desarrollo de aplicaciones multi-agente sobre dispositivos móviles con JME

BESA/ME: multi-agent applications development platform over mobile devices with JME

Juan Pablo Garzón Ruiz, MSc. y Enrique Gonzalez Guerrero, PhD.
Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana , Bogotá, Colombia.
jpgarzonruiz@hotmail.com, {jpgarzon, egonzal } @javeriana.edu.co

Recibido para revisión 29 de Mayo de 2009, aceptado 23 de Octubre de 2009, versión final 30 de Noviembrede 2009

Resumen—Este artículo presenta una descripción de la arquitectura del contenedor para Sistemas MultiAgentes BESA (Behavior-oriented, Event-driven and Social-based Agent Framework), orientado al desarrollo de aplicaciones que involucren principalmente teléfonos inteligentes (smartphone's). El desarrollo de esta implementación se basa en las características de programación ofrecidas por la arquitectura JME[10] debido a que la mayoría de empresas que elaboran teléfonos inteligentes lo utilizan como estándar de facto, pretendiendo así, ampliar el campo de acción e investigación para aplicaciones basadas en Sistemas MultiAgentes y crear software con dispositivos de gran difusión.
El desarrollo utilizando JME, permite ampliar su uso a diferentes plataformas de hardware y software en un entorno restrictivo con la instalación de los siguientes componentes: Configuración para Dispositivos con Limitada Conectividad (CLDC) [11] que incluye la maquina virtual (KVM) [12] y el perfil para dispositivos con Información Multimedia (MIDP) [13]; MIDP permite, entre otras ventajas, conectividad inalámbrica con diferentes tecnologías como Wi-fi [14], Bluetooth[15] e IRDA[16] principalmente, manejo de un entorno gráfico de alto nivel, persistencia (RMS) [17] y administración del ciclo de vida de la aplicación.

Palabras Clave—Sistemas Distribuidos, Sistemas multi-agente, Plataformas para Desarrollo de Sistemas MultiAgentes, BESA, JME, Teléfonos Inteligentes.

Abstract—This paper presents a description of the container for multi-agent systems' architecture BESA (Behavior-oriented, Event-driven and Social-based Agent Framework), designed to develop applications involving smart phones. The development of this implementation is based on the programming features offered by JME architecture, due to most companies that make smart phones use JME as a facto standard, pretending expand the scope and research to applications based on multi-agent systems and create software oriented to devices with wide diffusion.
The development using JME, allows to expand its use to different hardware and software platforms in a restrictive environment with the installation of the following components: Configuration for Devices with Limited Connectivity (CLDC), which includes the virtual machine (KVM) and the profile for devices Information with Multimedia (MIDP). MIDP allows, among other advantages, wireless connectivity with different technologies like Wi-Fi, Bluetooth and IRDA, managing a high-level graphical environment, persistence (RMS) and management of application's lifecycle.

Keywords—Systems, multi-agent Systems, BESA, JME, SmartPhones.

I. INTRODUCCION

El creciente avance tecnológico en campos de investigación, comerciales y académicos, hace obligatorio que tanto investigadores, docentes, estudiantes como programadores permanezcan en una constante búsqueda de nuevas tecnologías y mecanismos que permitan no solo suplir, sino superar, los requerimientos del entorno actual. Es en esta necesidad donde los SMA, permiten lograr dicho objetivo, mediante la generación de un nuevo paradigma encaminado hacia la programación orientada a agentes en sistemas distribuidos; logrando el desarrollo de aplicaciones y la implementación de sistemas robustos bajo diseños más simples y efectivos.

Se han propuesto diversas aproximaciones y soluciones para el desarrollo de sistemas basados en agentes. Estas aproximaciones están orientadas a proveer los mecanismos necesarios para hacer que los SMA puedan implementarse más fácilmente, buscando que sean confiables, escalables y sobre todo útiles en cualquier campo donde se deseen emplear. Una forma de clasificar las plataformas de agentes incluye las siguientes categorías: plataformas de concurrencia y seguridad, plataformas orientadas a inteligencia artificial y plataformas compatibles con el estándar FIPA [4] (Foundation for Intelligent Physical Agents). La primera categoría incluye aquellas plataformas cuyo principal objetivo es ofrecer mecanismos explícitos de seguridad, concurrencia y verificación de SMA; por ejemplo, Helsinger [8] propone PSE que busca establecer una plataforma de control multicapas con métricas observables

de alto nivel y las acciones de control de bajo nivel; por su parte, Welch [9] utiliza los principios de CSP para la fase de diseño, garantizando que el sistema resultante no poseerá problemas de abrazos mortales o similares. La segunda, abarca todas aquellas plataformas que están orientadas al desarrollo de agentes inteligentes que hacen explícito el manejo del conocimiento; por ejemplo, Beneventano [2] propone la plataforma MIKS, la cual maneja la integración y consulta de múltiples y heterogéneas fuentes de información; Cabri propone la plataforma BRAIN[3], en donde las interacciones ente los agentes están basadas en el concepto de rol. La última categoría agrupa las plataformas que aplican el modelo propuesto por FIPA y ofrecen un conjunto de clases base para la construcción de SMA; la plataforma más conocida de este tipo es JADE [1], la cual proporciona a los agentes la capacidad de descubrimiento dinámico de otros agentes y la comunicación entre ellos de acuerdo al paradigma P2P.

En el grupo de investigación SIDRe se ha desarrollado una plataforma de agentes denominada BESA [5], la cual partiendo desde la perspectiva de una plataforma tipo FIPA, integra mecanismos fuertes para el manejo del paralelismo y concurrencia. Actualmente, se busca el desarrollo de nuevas herramientas que amplíen el campo de acción de las aplicaciones para SMA que pueden ser construidas con la plataforma BESA. El modelo conceptual BESA [6] es genérico y permite la construcción de plataformas de agentes que permitan la operación de SMA en diferentes ambientes heterogéneos. Para lograr un mayor cubrimiento en el desarrollo de aplicaciones, tanto a nivel académico como a nivel organizacional, se continua con la generación de más entornos que gracias al diseño de mecanismos de interoperabilidad permitirán la interacción entre contenedores BESA heterogéneos.

En este artículo se presentan dos contribuciones importantes. Por una parte, el desarrollo de la plataforma BESA para ambientes JME. Por otra parte, la generación de un modelo de interoperabilidad entre contenedores de diferente naturaleza, el cual además permite alcanzar un mayor cubrimiento geográfico de las aplicaciones distribuidas desarrolladas aplicando el modelo BESA.

La implementación de este diseño basa su esencia en los requerimientos de adaptabilidad del primer modelo desarrollado bajo Java y su equivalencia funcional bajo el entorno de programación brindado por JME. Esta implementación conserva el modelo de servicios del estándar FIPA.

En este artículo, antes de presentar el desarrollo de BESA JME, se introducen los conceptos fundamentales del modelo BESA, la arquitectura general y su implementación. También, se explica el modelo de interoperabilidad propuesto, con el cual se garantiza la transparencia de comunicación entre agentes de plataformas heterogéneas o dispersas geográficamente. Finalmente se presenta una solución a los problemas inherentes a los dispositivos móviles.

II. CONCEPTOS FUNDAMENTALES

El modelo abstracto de BESA se basa en tres conceptos fundamentales: una arquitectura orientada a comportamientos del agente, una técnica de control orientada a eventos que implementa un mecanismo de manejo de la concurrencia tipo select, y un soporte basado en mecanismos sociales para facilitar la cooperación entre los agentes. Estos conceptos han sido presentados en detalle en [5], [6] y [7], en esta sección se explican en forma más breve.

A. Orientado a Comportamientos (Behavior-Oriented)

Con el fin de buscar la reducción de complejidad de los SMA se toma como idea fundamental el dividir una entidad compleja en un conjunto de entidades más simples, brindándoles a estas la capacidad para dar realización a un propósito bien definido. Para este fin, un agente BESA se compone de un conjunto de comportamientos concurrentes, módulos paralelos encargados de responder a un conjunto bien definido de eventos.

B. Manejo de Eventos (Event-Driven)

En el modelo BESA, un evento se puede interpretar como una señal que puede ser percibida o usada por un agente que está interesado en ella, la cual puede incluir información sobre lo qué ha sucedido en el ambiente. El manejo de eventos permite tener una gran reactividad, tan pronto ocurre un evento el comportamiento o comportamientos asociados se activan para que ejecuten la función o funciones de tratamiento asociadas. Además, la orientación a eventos evita la aparición de esperas ocupadas que generen desperdicio en el uso de los recursos de cómputo disponibles.

Para el manejo del no-determinismo temporal de los eventos, BESA incluye mecanismos tipo select (inspirados de los lenguajes concurrentes) para el manejo de los mismos. Estos mecanismos se basan en el modelo de un sistema de guardas. Cuando la guarda se dispara el tratamiento asociado se ejecuta. Una guarda es disparada, si al verificar una expresión booleana está resulta verdadera y el evento de comunicación asociado a la guarda ha ocurrido. Generalmente, la expresión booleana se relaciona con el estado de la entidad que se está procesando, y el evento de comunicación es asociado a la recepción de un tipo predeterminado de mensaje. Una vez la guarda se dispara el evento es transferido a los comportamientos que deben realizar su tratamiento.

C. Basado en lo Social (Social-Based)

Para analizar y diseñar un SMA, el uso de un modelo basado en lo social permite estudiar el sistema como una organización de entidades que interactúan entre sí.

En un modelo basado en lo social, una organización puede ser considerada como un conjunto de organizaciones de bajo nivel. Esta descomposición recurrente de la organización en más simples, proporciona una arquitectura para diseñar SMA. En el nivel más alto de abstracción el SMA se descompone en

un conjunto de organizaciones con una semántica y funciones bien definidas, estas organizaciones son subdivididas nuevamente de una manera recurrente, finalmente entidades asimiladas a los agentes son obtenidas. Los agentes que resultan tienen una semántica clara, que incluye una meta en la organización y una interfaz bien definida con otros agentes y el ambiente. Para facilitar este tipo de construcciones, en BESA se incluye una capa con servicios que facilitan la implementación de protocolos de interacción entre agentes.

III. ARQUITECTURABESA

La arquitectura BESA está compuesta por tres niveles: nivel agente, nivel social y nivel de sistema [5][6][7].

A. Nivel Agente

La arquitectura interna de un agente integra dos características importantes: una composición modular de comportamientos y un mecanismo selector de eventos. Un agente está compuesto por un canal, un conjunto de comportamientos y un estado. El estado es una memoria compartida accesible a los comportamientos del agente usando sincronización de exclusión mutua. En la figura 1 se presentan los componentes del agente y la manera como interactúan.

El canal recibe los eventos direccionados hacia el agente e implementa un esquema selector basado en guardas. Un canal posee un único punto de entrada para los eventos, con lo cual se puede garantizar que la semántica del mecanismo selector es respetada. Todos los eventos, incluso los internos, se reciben en un buzón y se deben procesar por el selector de guardas. El selector de guardas transfiere los eventos entrantes al puerto apropiado; la selección del puerto se basa en el tipo del evento. Una guarda dispara un mecanismo asincrónico, que verifica si ésta puede ser disparada cuando un evento llega o cuando una variable que influye en la condición booleana de la guarda se modifica.

Cuando una guarda es disparada, el primer evento en el puerto se envía a los comportamientos apropiados. Los comportamientos son procesos paralelos que trabajan juntos para lograr las metas del agente. Un comportamiento permite reaccionar a un conjunto bien definido de tipos de eventos; para demostrar su interés en un tipo específico de evento. El comportamiento es registrado ante la guarda asociada al evento; un evento le será enviado cuando la guarda sea disparada.

Cada comportamiento tiene una cola para recibir eventos de los puertos del canal. Cuando llega un evento, el comportamiento ejecuta automáticamente el procedimiento de tratamiento asociado a la guarda. Este procedimiento incluye el programa apropiado para reaccionar y para producir una respuesta al evento recibido.

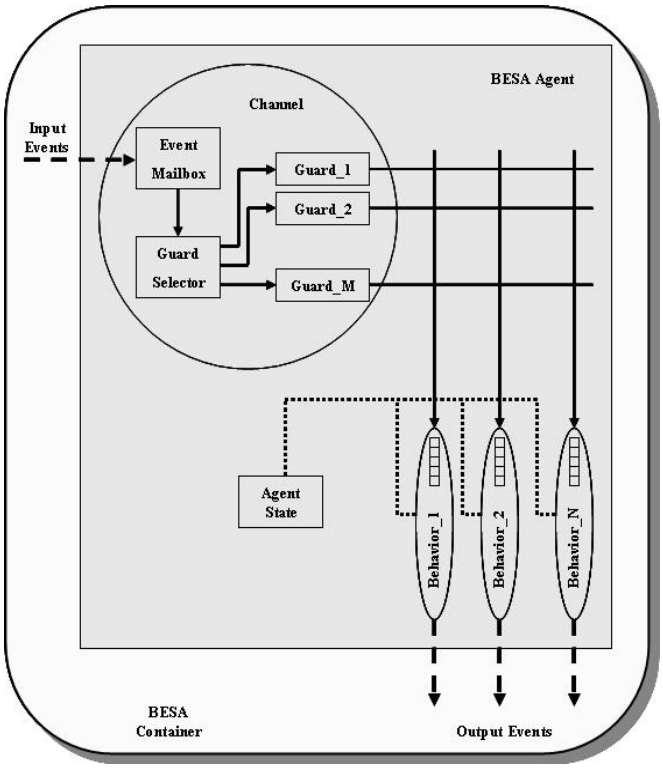

Figura 1. Arquitectura interna de un agente.

Una guarda tiene dos procedimientos asociados. El primero es ejecutado automáticamente por el mecanismo de disparo para verificar la condición booleana. El segundo es ejecutado por un comportamiento cuando la guarda ha sido disparada.

B. Nivel Social

En el modelo abstracto de BESA, los niveles intermedios de abstracción son incorporados para permitir la definición y manipulación de organizaciones jerárquicas. Una organización se define como un conjunto de agentes asociados a un agente mediador del siguiente nivel más alto de la jerarquía global de la organización. El mediador actúa como facilitador del trabajo cooperativo de los agentes en la organización.

El agente mediador puede servir como un despachador de eventos que recibe eventos y los reenvía a los agentes en cuestión. Puede también servir como puente a un conjunto de los agentes capaces de ejecutar o proveer un servicio; en este caso el mediador es registrado en un directorio de páginas amarillas como el prestador de un servicio, cuando un evento de la petición llega a él, el evento es direccionado al agente apropiado. En una situación más compleja, el mediador puede ser diseñado para manejar mecanismos de cooperación, proporcionando los servicios para: asignación de tareas y recursos; resolución de conflictos; y planeación y coordinación distribuidas.

En BESA, la implementación de las relaciones sociales es soportada por el mecanismo selector interno de cada agente.

Un servicio del mediador se puede modelar e implementar por un protocolo de interacción, que es una secuencia predeterminada de interacciones, intercambio de eventos, entre los agentes involucrados. Una interacción se puede modelar por un evento bien definido, que se puede asociar a una guarda. Cada agente involucrado en el protocolo debe incluir un comportamiento para manejar la secuencia de interacciones requeridas. Para el diseñador del sistema, el trabajo se reduce a proporcionar los procedimientos asociados a un conjunto predefinido de guardas; la semántica y los datos de estas guardas son determinados por su protocolo de interacción asociado.

C. Nivel de Sistema

Un sistema de agentes implementado usando la arquitectura BESA se considera como sistema distribuido compuesto por uno o varios contenedores BESA, que pueden funcionar en diversas máquinas físicas o virtuales. El sistema entero incluye un único puerto compatible con FIPA, que recibe la comunicación y las consultas de otros sistemas externos usando FIPA ACL.

Un contenedor BESA es un espacio de ejecución donde los agentes habitan; este maneja su ciclo vital. Un contenedor tiene un administrador local, que apoya la comunicación del agente y proporciona servicios de directorios de páginas blancas y amarillas. Los contenedores de un sistema BESA funcionan juntos e implementan un mecanismo de replicación que hace transparente la identificación de agentes, la localización y la migración entre contenedores. Cuando una nueva instancia de un agente se crea, el administrador local del contenedor lo registra en el directorio de páginas blancas. Si se requiere, un agente puede colocar los servicios que proporciona en el directorio de páginas amarillas. Cuando un cambio en el ciclo de vida de un agente se realiza o cualquiera de los directorios se modifica, un mecanismo de replicación se activa para mantener la consistencia de todos los administradores locales.

El directorio de páginas blancas está compuesto por una colección de manijas de identificación de los agentes. Una manija incluye toda la información necesaria para localizar un agente en el sistema. Cuando un agente requiere enviar un evento a otro agente, realiza una solicitud al administrador local para que le retorne la manija del agente destinatario, una vez obtenida la manija, se realiza el envío del evento de acuerdo al mecanismo asociado a la manija.

IV. IMPLEMENTACIÓN BESA/ME [17]

Basado en el modelo abstracto de BESA y su actual implementación desarrollada en Java, se definen requerimientos particulares para poder realizar la implementación de la plataforma de acuerdo a la arquitectura del modelo propuesto.

En general, todos los modelos de clases y de sincronización utilizados en la versión Java se mantuvieron para la implementación en JME. Por tanto la construcción de la plataforma se centró en buscar los mecanismos que permitieran obtener el mismo modelo operativo que ya había sido ampliamente validado pero con restricciones inherentes a dispositivos móviles como: poca capacidad de procesamiento, almacenamiento e intermitencia en la comunicación. Esta aproximación garantiza la fiabilidad de la nueva plataforma.

A nivel general, la arquitectura está conformada por dos entornos, el entorno dinámico DE y el entorno estático SE, que se comunican a través de un acceso inalámbrico WA. En la figura 2 ilustra la división en los dos entornos, como los dispositivos que se encuentran en ellos.

El entorno dinámico DE se entiende como el conjunto de dispositivos móviles MD que pueden cambiar continuamente de ubicación geográfica. Estos dispositivos móviles MD tiene un acceso inalámbrico WA para comunicarse con el entorno estático SE el cual se denomina acceso inalámbrico móvil MWA.

El entorno estático SE es una agrupación de dispositivos estáticos SD, los cuales incluyen computadores personales, servidores, entre otros que no cambian de ubicación física, y de dispositivos livianos WSD. Los SD y los WSD están comunicados entre ellos a través de una red LAN o WAN.

El acceso inalámbrico móvil MWA tienen un área de cubrimiento que puede variar dependiendo de la tecnología y el protocolo de comunicación que se utilice entre los dos entornos. Al conjunto de estas áreas de cubrimiento se le conoce como ambiente físico cerrado. Las áreas de cubrimiento pueden superponerse entre ellas, lo cual no es obligatorio.

En la figura 2 todos los dispositivos móviles MD utilizan wi-fi por lo cual tienen un alcance de aproximadamente 100 metros en un ambiente libre de obstáculos. El dispositivo móvil MD de la WLAN de la izquierda se encuentra dentro del área de cubrimiento del otro dispositivo móvil permitiendo su interconectividad inalámbrica; los dos dispositivos móviles de la WLAN del centro tienen acceso inalámbrico ampliado haciendo posible de forma transparente y permanente la conectividad entre las WLAN del centro y de la derecha cubriendo una mayor zona geográfica, lo mismo sucede con el dispositivo móvil de la WLAN de la derecha. Independiente de la WLAN donde se encuentra los dispositivos cada uno de ellos por medio de los puntos de acceso AP, tienen conectividad con los dispositivos estáticos SD, al igual que comunicación con el dispositivo liviano WSD. La WLAN superpuesta se conoce como red inalámbrica local ampliada WWLAN [14].

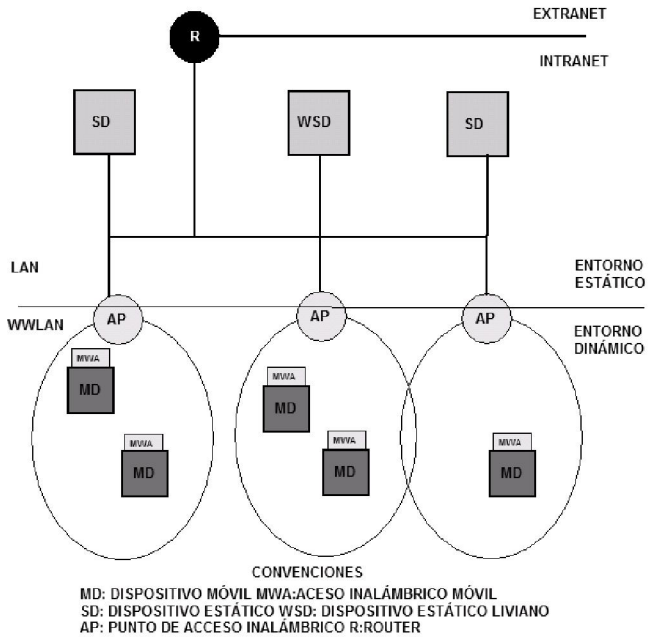

Figura 2. Vista general de la arquitectura

A. Nivel Agente

La arquitectura en un entorno JME está conformada por cuatro agentes:

Peer: este agente depende de la aplicación APP, por lo que es decisión del desarrollador determinar cómo se implementa. Puede ser móvil o estático y es el encargado de procesar las solicitudes del usuario.

APS: este agente es el único punto de entrada y salida de una aplicación APP. Es el encargado de solicitar y prestar servicios entre aplicaciones APP.

AM: este agente es el encargado de almacenar y recuperar los artefactos en el repositorio.

SSP: este agente es el encargado de invocar los servicios para acceder a los diferentes recursos ya sea un Web Service, una Base de Datos, etc.

La plataforma provee una implementación de las funcionalidades básicas de cada agente. Esta implementación se hereda y cada desarrollador puede extender su funcionamiento según las necesidades de la aplicación APP que esté desarrollando.

A continuación se presenta el modelo de interacción existente en el agente peer y los otros tres se explican debido a sus características en el nivel social.

A1. Agente Peer

A continuación se plantea el análisis de las funcionalidades básicas de este agente.

a. Estado
1. Referencia al agente APS.
2. Password de aplicación a la que pertenece.
3. Identificador de la aplicación a la que pertenece
4. Referencia al directorio de páginas amarillas y blancas.
5. Propio de cada aplicación.

b. Metas

M11. Enrutar artefacto para almacenar en repositorio.

M12. Recuperar artefacto almacenado previamente en un repositorio.

M13. Registrarse en el servidor de punto de acceso APS.

M14. Solicitar servicio prestado por otra aplicación APP.

M15 Solicitar servicio de operaciones CRUD (create, read, update, delete) sobre un recurso.

M16. Prestar servicios a otras aplicaciones APP.

c. Entradas Sensoriales

i. Asincrónicas

No hay.

ii. Sincrónicas

S.1.1. Encontrar al agente AM en las páginas blancas y amarillas.

S.1.2. Encontrar al agente SP en las páginas blancas y amarillas.

S.1.3. Encontrar al agente APS en las páginas blancas y amarillas.

S.1.4. Encontrar a otro agente Peer en las páginas blancas y amarillas.

d. Actuadores

S.1.5. Registrarse en las páginas amarillas y blancas.

e. Comportamientos

B1.1. Almacenar artefacto. Cumple con la meta M11. La interacción se representa en la figura 3.

Mensaje / Evento: Artefact_Store_Reply, devolución de un mensaje que contiene si la acción se pudo realizar o no y el comprobante que identifica al artefacto que ha sido almacenado, la aplicación del dispositivo móvil a la que pertenece, el id del AM y la fecha de almacenamiento.

Acción: El agente almacena en una tabla el comprobante recibido para poder recuperarlo en cualquier momento.

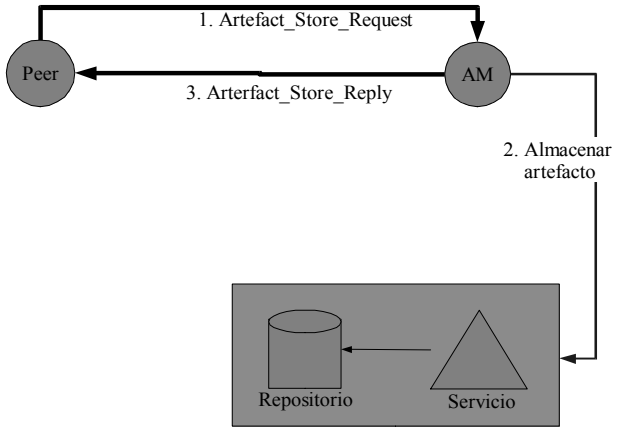

Figura 3. Almacenar artefacto.

B1.2. Recuperar artefacto. Cumple con la meta M12. La interacción se representa en la figura 4.

Mensaje / Evento: Arterfact_Retrieve_Reply, se le devuelve el artefacto solicitado que se encontraba almacenado en el repositorio. Si se presenta una excepción, se devuelve la notificación correspondiente

Acción: Almacena el artefacto mientras espera una instrucción final.

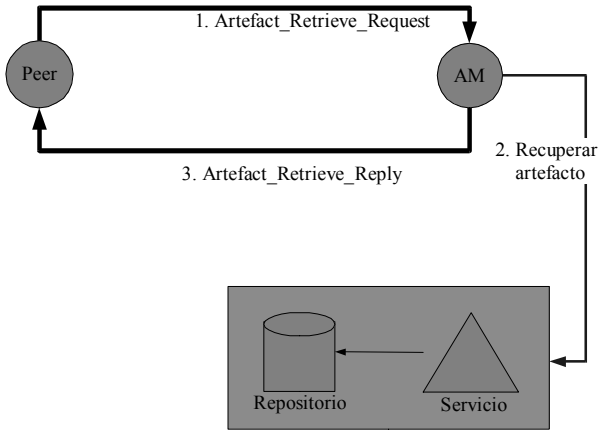

Figura 4. Recuperar artefacto.

B1.3. Solicitar servicio. Cumple con la meta M15. La interacción se representa en la figura 5.

Mensaje / Evento: Service_Reply, se le devuelve la respuesta de acuerdo al servicio que solicitó. Si ocurre alguna excepción, se le retorna una notificación de lo sucedido.

Acción: El desarrollador de la aplicación establece que se debe realizar de acuerdo a la respuesta obtenida y al objetivo de la aplicación APP.

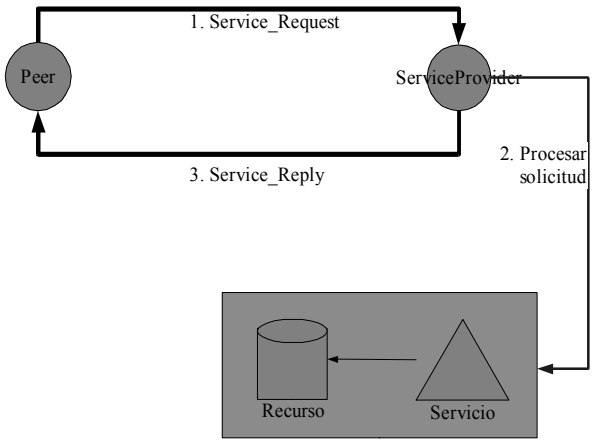

Figura 5. Solicitar servicio.

B1.4. Registrar servicio. Cumple M13. La interacción se representa en la figura 6.

Mensaje / Evento: Service_Register_Reply, devuelve un mensaje de ACK indicando si el registro se realizó satisfactoriamente o ha fallado.

Acción: El desarrollador de la aplicación establece que se debe realizar dependiendo del contenido del mensaje de ACK que reciba.

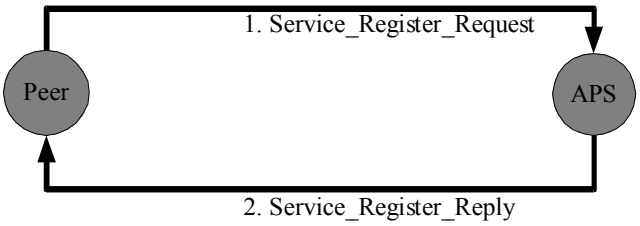

Figura 6. Registrar servicio.

B1.5. Prestar servicio a otra aplicación APP. Cumple M14. La interacción se representa en la figura 7.

Mensaje / Evento: Service_A_Request, se le envía un mensaje indicándole que servicio se requiere y los parámetros necesarios para llevarlo a cabo. La estructura que maneja estos parámetros la define el desarrollador.

Acción: el Peer ejecuta el servicio solicitado de acuerdo al código programado por el desarrollador.

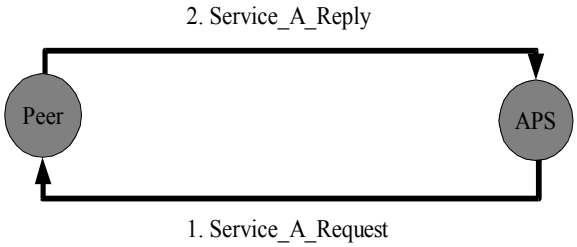

Figura 7. Prestar servicio a otra aplicación APP.

B1.6. Solicitar servicio de otra aplicación APP. Cumple M14. La interacción se representa en la figura 8.

Mensaje / Evento: Service_APP_Reply, devuelve un mensaje con la respuesta del servicio solicitado. Este mensaje puede ser de acuse de recibo ACK o con información dependiendo de cómo el desarrollador haya implementado el servicio.

Acción: El desarrollador de la aplicación establece que se debe realizar dependiendo del contenido del mensaje que le llegue.

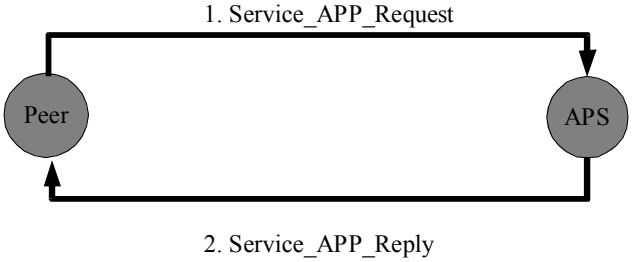

Figura 8. Solicitar servicio de otra aplicación.

B. Nivel Social

Como se definió antes en el modelo abstracto de BESA, los niveles intermedios de abstracción son incorporados para permitir la definición y manipulación de organizaciones jerárquicas. Una organización se define como un conjunto de agentes asociados a un agente mediador del siguiente nivel más alto de la jerarquía global de la organización. El mediador actúa como facilitador del trabajo cooperativo de los agentes en la organización.

Es así como desde el concepto de agrupación de agentes que utiliza un mediador en la plataforma se tienen tres agentes especiales. El agente APS que es el punto de entrada y salida de una aplicación y el agente AM que es el encargado de almacenar y recuperar los artefactos en el repositorio

A1. Agente AM

a. Estado

1. Referencia al directorio de páginas amarillas

2. Referencia al servicio S que puede acceder para el manejo del repositorio.

b. Metas

M11. Almacenar artefacto.

M12. Recuperar artefacto.

c. Entradas Sensoriales

Asincrónicas

No hay.

Sincrónicas

No hay.

d. Actuadores

S.1.1. Registrarse en las páginas amarillas y blancas.

e. Comportamientos

B1.1. Almacenar artefacto. Cumple con M31. La interacción se representa en la figura 9.

Mensaje / Evento: Artefact_Store_Request, solicitud de almacenamiento de un artefacto en el repositorio. La solicitud está compuesta por un mensaje que indica el identificador de la aplicación dueña del artefacto, el tiempo que debe ser almacenado el artefacto y el artefacto para almacenar.

Acción: El AM hace una invocación al servicio encargado de almacenar el artefacto en el repositorio, enviándole el artefacto, el id de la aplicación, su id (AM) y el tiempo que debe ser almacenado. Cuando se realiza el almacenamiento, el servicio le devuelve un comprobante que contiene el identificador de la aplicación que envió el artefacto, el identificador único del artefacto, el id del AM que solicitó el servicio y la fecha de almacenamiento. El servicio almacena el comprobante en una tabla.

Respuesta: Se envía un comprobante hacia el peer P que envió la solicitud.

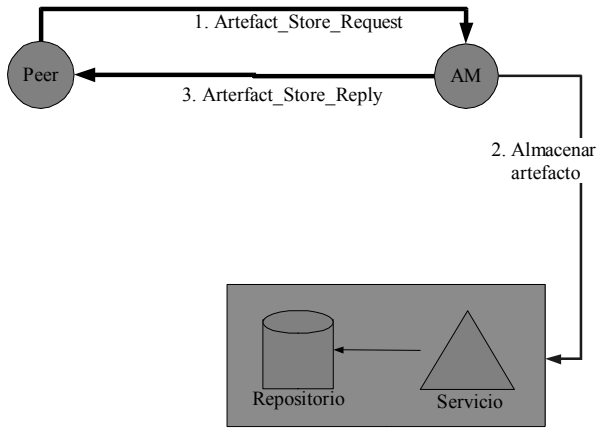

Figura 9. Almacenar artefacto

B1.2. Recuperar artefacto. Cumple con M32. La interacción se representa en la figura 10.

Mensaje/ Evento: Arterfact_Retrieve_Request, solicitud para recuperar un artefacto que se encuentra en el repositorio. La solicitud está compuesta por un mensaje que indica el identificador de la aplicación que solicita el artefacto y el comprobante

Acción: El AM verifica que el identificador de la aplicación que solicita el artefacto y la aplicación autorizada para recuperar el artefacto según el comprobante sea el mismo. Si esto se cumple, el AM hace una invocación al servicio encargado de recuperar el artefacto en el repositorio, enviándole el comprobante.

Respuesta: Se devuelve el artefacto al Peer que lo solicitó. Si surgió alguna excepción porque la aplicación que solicitaba el artefacto no estaba autorizada para consultarlo o porque el tiempo de almacenamiento se había excedido, se envía la notificación respectiva.

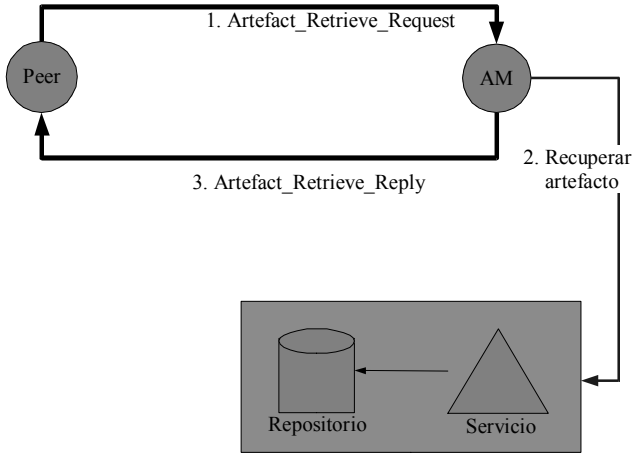

Figura 10. Recuperar artefacto.

A.2. Agente APS
a. Estado
1. Referencia al directorio de páginas amarillas
2. Tabla de usuarios y passwords autorizados para utilizar los servicios que presta la aplicación.
3. Tabla de servicios prestados por la aplicación y el Peer que presta el servicio.
b. Metas
M21. Registrar servicio.
M22. Solicitar servicio a otra aplicación.
M23. Prestar servicio a otra aplicación.
c. Entradas Sensoriales
i. Asincrónicas
S.2.1. No hay.
ii. Sincrónicas
S.2.2. Consultar páginas amarillas y páginas blancas.
d. Actuadores
S.2.3. Registrarse en las páginas amarillas
e. Comportamientos
B2.1. Registrar servicio. Cumple con M41. La interacción se representa en la figura 11.

Mensaje / Evento: Service_Register_Request, solicitud de registro del servicio que presta Peer a otra aplicación. El mensaje está compuesto por el identificador del peer, el identificador del servicio y los parámetros necesarios para llevar a cabo el servicio.

Acción: El APS registra en su tabla el identificador del peer, el identificador del servicio y los parámetros necesarios para llevar a cabo el servicio.

Respuesta: Se envía un acuse de recibo hacia el Peer que envió la solicitud de registro indicándole si fue exitosa o no.

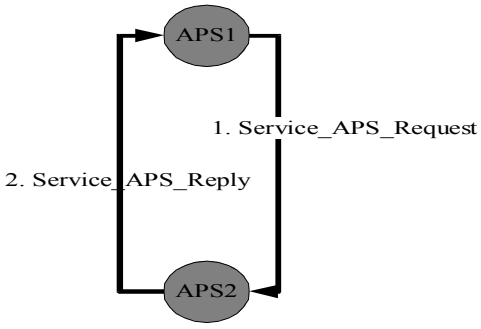

Figura 11. Registrar servicio.

B2.2. Solicitud de un servicio entre aplicaciones APP. Cumple con M42 y M43. La interacción se representa en la figura 12.

Mensaje / Evento: Service_APS_Request, solicitud de consulta de un servicio que ofrece la otra aplicación APP. El mensaje está compuesto por el usuario y password, el identificador del servicio y los parámetros necesarios para llevar a cabo el servicio.

Acción: El APS verifica en su tabla de usuarios y password que estos sean correctos. Posteriormente, busca en su tabla de prestación de servicios y realiza la solicitud al peer correspondiente enviándole el identificador del servicio y los parámetros necesarios para llevar a cabo el servicio.

Respuesta: Si el usuario y password no son válidos se retorna un mensaje de error.

Si son válidos el usuario y el password, se retorna el resultado enviado por el peer que ejecute la solicitud.

Mensaje / Evento: Service_APS_Reply, resultado que obtuvo el APS al solicitar el servicio a la otra aplicación APP.

Acción: El APS reenvía el resultado al peer que le solicitó el servicio.

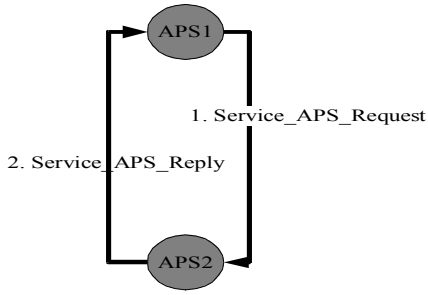

Figura 12. Solicitud de un servicio entre aplicaciones.

B2.3. Solicitud servicio de un peer a otra aplicación APP. Cumple con M43. La interacción se representa en la figura 13.

Mensaje / Evento: Service_APP_Request, solicitud de consultar un servicio en otra aplicación APP. La solicitud está compuesta por el identificador del servicio y los datos necesarios para ejecutar el servicio.

Acción: El APS busca en las páginas amarillas que aplicación APP presta ese servicio y reenvía la solicitud.

Respuesta: Se envía un mensaje con la respuesta del servicio solicitado. Si por alguna razón el servicio no se pudo llevar a cabo satisfactoriamente, se envía una excepción.

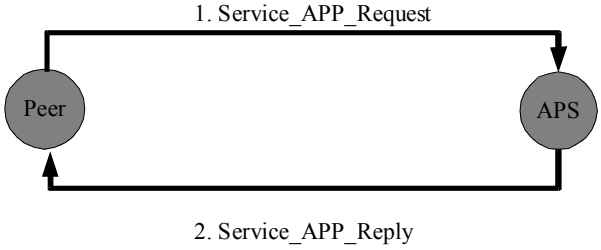

Figura 13. Solicitud servicio de un peer a otra aplicación.

B2.4. Solicitud por parte del APS a un servicio de un peer. Cumple con M43. La interacción se representa en la figura 14.

Mensaje / Evento: Service_A_Reply, solicitud que contiene la respuesta del servicio solicitado al Peer.

Acción: El APS reenvía la respuesta al APS que le solicitó el servicio.

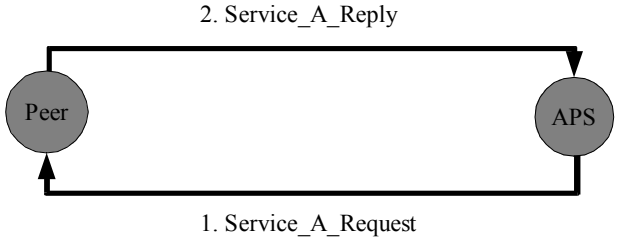

Figura 14. Solicitud por parte del APS a un servicio de un peer.

A3. Agente SP

a. Estado

1. Referencia al directorio de páginas amarillas

2. Referencia a los servicios S que puede acceder.

b. Metas

M31. Solicitar operación CRUD (create, read, update, delete).

c. Entradas Sensoriales

Asincrónicas

No hay.

Sincrónicas

No hay.

d. Actuadores

S.3.1. Registrarse en las páginas amarillas y blancas.

e. Comportamientos

B3.1. Solicitar servicio. Cumple con M21. La interacción se representa en la figura 15.

Mensaje / Evento: Service_Request, solicitud de realizar una operación CRUD en un recurso. La solicitud está compuesta por el tipo de operación que se desea realizar, y los datos necesarios para poder llevar a cabo la operación.

Acción: El SSP hace una invocación al servicio encargado de realizar la operación solicitada, enviándole el tipo de operación y los datos necesarios.

Respuesta: Se envía un mensaje con la respuesta del servicio solicitado al Peer que lo invocó. Si por alguna razón el servicio no se pudo llevar a cabo satisfactoriamente, se envía una excepción.

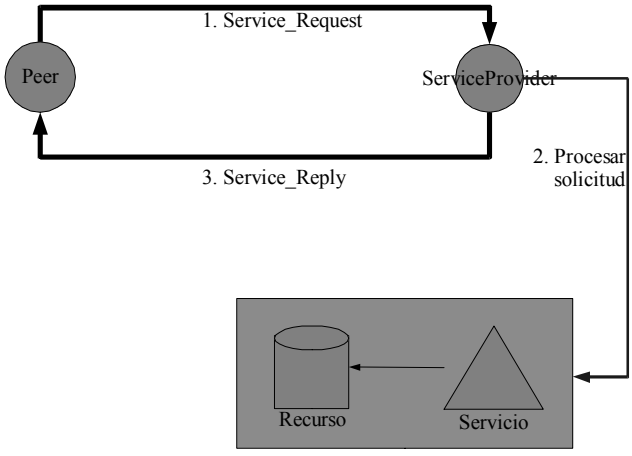

Figura 15. Solicitar servicio.

C. Nivel de Sistema

Un sistema de agentes implementado usando la arquitectura BESA se considera como sistema distribuido compuesto por uno o varios contenedores BESA, que pueden funcionar en diversas máquinas físicas o virtuales. El sistema entero incluye un único puerto compatible con FIPA, que recibe la comunicación y las consultas de otros sistemas externos usando FIPA ACL.

1 Modelo Físico

En el modelo físico se muestra la distribución de los dispositivos tanto en el entorno estático SE como en el entorno dinámico DE. La figura 16 ilustra y permite observar de una manera clara cómo está distribuido.

Existen dos usuarios, A y B, con sus respectivos dispositivos móviles MD. Se encuentran distribuidos en el entorno dinámico DE y acaban de ingresar al ambiente cerrado de una aplicación.

En el entorno estático SE se encuentran localizados dos dispositivos estáticos SD, que se comunican entre ellos a través del protocolo TCP/IP. Existen dos servidores, Re, que tienen una base de datos donde se encuentra la información que va a consultar la aplicación APP.

2 Modelo Lógico

En el modelo lógico se muestra la distribución de los servicios S, los peers P y de las aplicaciones APP. La figura 17 ilustra la distribución de estas entidades.

Hay dos aplicaciones APP, corriendo dentro del ambiente cerrado. Una Aplicación APP1 tiene dos peers móviles MP ubicados en la WLAN del centro y tres peers estáticos SP ubicados en la LAN de la izquierda. La AP2 tiene tres peers móviles MP ubicados en la WLAN de la derecha y dos peers estáticos SP ubicados en la LAN de la derecha.

Hay dos servicios, S1 y S2 que acceden a los R1 y R2 respectivamente. Hay dos proveedores de servicios estáticos SSP1 Y SSP2, que utilizan los SP para acceder a la información de los Recursos.

3 Integración Modelo Físico y Modelo Lógico

La integración del modelo físico y lógico se ilustra en la figura 18.

Los MP de la APP1 se encuentran en el dispositivo móvil del usuario A, MD1 ubicado en la WLAN de la izquierda. Por su parte, los MP de la APP2 se encuentran en el MD del usuario B, MD2 ubicado en la WLAN de la derecha. A diferencia de los MP, los SP de una APP se encuentran distribuidos en varios dispositivos. Dos de los SP de la APP1 se encuentran en el SD ubicado a la izquierda y uno en el SD ubicado a la derecha. La APP2 tiene sus dos SP en el SD ubicado a la izquierda.

Los servicios S se encuentran en el servidor de base de datos al que están accediendo. El SSP1 se encuentra en el SD ubicado a la izquierda y el SSP2 en el SD ubicado a la derecha y pueden acceder a cualquier S.

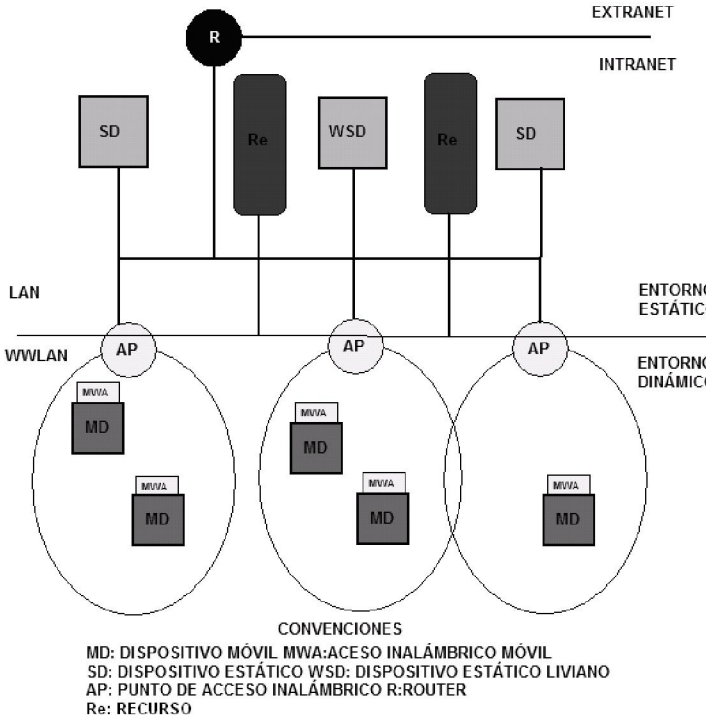

Figura 16. Modelo Físico

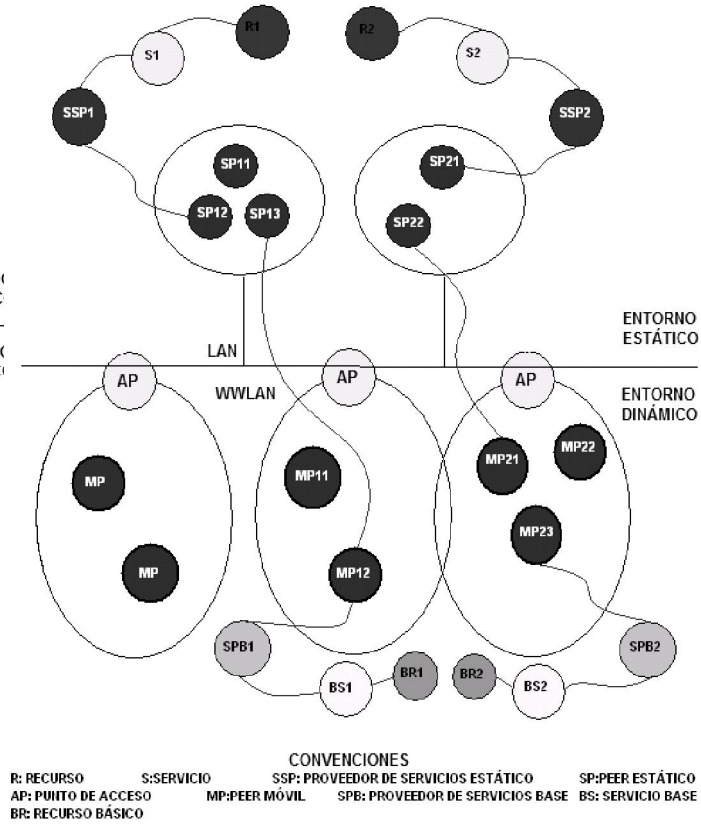

Figura 17. Modelo Lógico

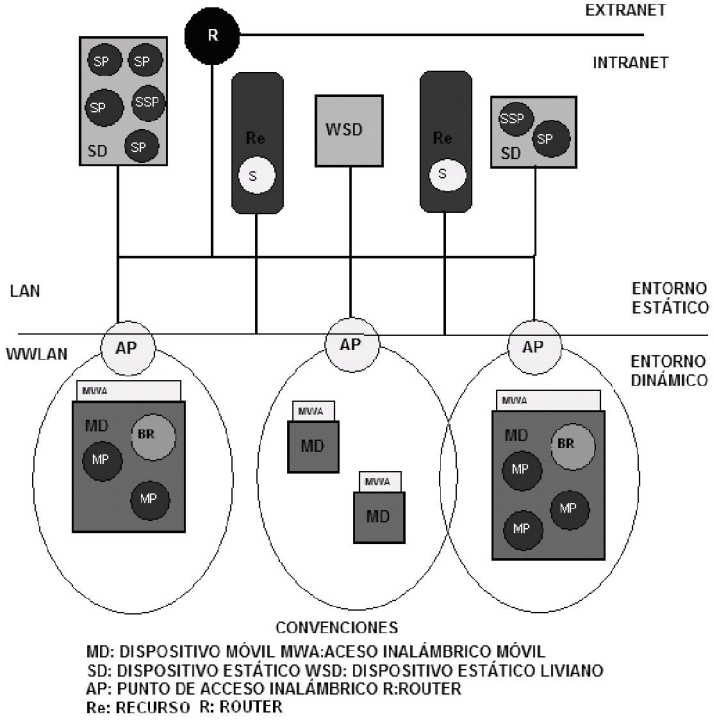

Figura 18. Modelo Integración.

V. SOLUCIÓN DE LAS RESTRICCIONES INHERENTES DE LOS DISPOSITVOS MÒVILES [17]

Durante el proceso de análisis y diseño de la arquitectura se tuvieron en cuenta varias posibilidades que atacaban una o varias de las limitaciones de los dispositivos móviles. Estas ideas fueron el punto de partida para la propuesta final.

A continuación se enumeran las propuestas más relevantes clasificadas según la limitación que atacan.

A. Limitada capacidad de procesamiento

Para enfrentar la reducida capacidad de procesamiento que tienen los dispositivos móviles se utilizaron las siguientes soluciones.

Procesos en el entorno estático o dinámico.

Una APP es una aplicación que se define como un conjunto de procesos que colaboran entre sí para cumplir una meta en común. Parte de los procesos de la APP están en el entorno estático SE y la otra parte en un dispositivo móvil MD. Cuando un proceso que está en el entorno dinámico DE no puede desarrollar su funcionalidad de una manera óptima puede solicitarle a otro proceso del entorno estático SE que le ayude a procesar la información y le devuelva el resultado cuando esté listo.

Este concepto de aplicación APP distribuida entre el entorno estático y el dinámico es una parte importante de la arquitectura.

Los procesos del dispositivo móvil MD, al tener conectividad por medio de un punto de acceso inalámbrico con el ambiente estático SE, establecen una conexión con un dispositivo estático SD y crean imágenes suyas, llamadas representantes. El representante es el responsable de realizar el procesamiento pesado de las aplicaciones que requieren los procesos de la aplicación móvil. Cuando finaliza el procesamiento requerido, el representante envía el resultado al proceso del dispositivo móvil MD para que continúe con su funcionamiento. Este procedimiento sucede de una manera transparente para el usuario-programador. En caso de que el dispositivo móvil MD pierda la conexión, el representante entra a un estado pasivo de ejecución.

Servidores

Incluir sistemas de computo(servidores) en el ambiente estático SE con mejor capacidad de computo que la de un dispositivo móvil. En la práctica se utilizan agendas digitales con velocidad de reloj de 416 MHz como la Dell Axim X51V, por lo tanto lo que se consideran como servidores deben proporcionar mínimo una capacidad de velocidad de reloj de 3 veces mayor. Por lo tanto los procesos de la aplicación móvil que necesitan ejecutar un procesamiento pesado solicitan el servicio residente en servidores. Cuando el servidor ha ejecutado el trabajo solicitado, envía los datos y/o la respuesta obtenida al solicitante. Este tipo de solución es adecuada si se utiliza una aproximación distribuida con múltiples servidores, evitando de esta forma cuellos de botella o puntos críticos de falla.

Este concepto de servidores distribuidos que permiten un acceso controlado a los recursos y procesamiento por demanda es una parte importante de la arquitectura.

B. Reducida capacidad de almacenamiento

Dada la limitación de almacenamiento de los dispositivos móviles se utiliza un repositorio de datos distribuido.

El repositorio permite almacenar solamente artefactos que el dispositivo móvil no utilice en determinado momento. Las únicas operaciones permitidas sobre estos artefactos son guardar y recuperar. Si el dispositivo móvil propietario desea realizar alguna modificación sobre el artefacto debe recuperarlo, realizar la operación y después volver a almacenarlo. Por cada artefacto se tendrá un comprobante con un identificador único de artefacto, el identificador del dispositivo móvil propietario, fecha de depósito y tiempo máximo de almacenamiento. El repositorio de información utiliza un directorio donde lleva el registro de los comprobantes de los artefactos que tiene almacenados. Si una aplicación APP1 desea que otra aplicación APP2 recupere su artefacto, la APP1 le envía el comprobante para que así la APP2 pueda recuperarlo.

C. Reducida duración de energía e intermitencia en la comunicación

La intermitencia en la comunicación y la reducida duración de la energía, a pesar de ser dos limitaciones diferentes, tienen como consecuencia que el dispositivo móvil queda temporalmente sin conexión con el entorno estático. Por esta razón, se atacan de una manera similar. A continuación se explica la propuesta que surgió con respecto a este tema.

La arquitectura tiene un componente que funciona como un proxy, el cual es el encargado de verificar el estado del canal de comunicación entre los dispositivos del entorno estático SE y los del dinámico DE. Si la conexión de este canal se interrumpe, el proxy almacena temporalmente los mensajes que no se pueden enviar en una cola hasta que se restablezca la comunicación. Para el manejo de los mensajes almacenados en la cola surgen diferentes opciones:

Verificar periódicamente si el canal se ha restablecido, y en caso dado enviar todos los mensajes pendientes.

Disponer de un mecanismo asincrónico, soportado por el controlador del dispositivo de comunicación, que notifique cuando el canal se ha restablecido.

Adicionalmente a la detección del estado del canal, se puede incorporar un mecanismo de time-out a los mensajes. Después de cierto tiempo de espera se le notifica a la entidad que solicitó el envío del mensaje que no se pudo trasmitir por fallas en el canal de comunicación.

La solución consiste en disponer de un mecanismo que notifique cuando el canal se ha restablecido. Adicionalmente, se incorpora el mecanismo de time-out a los mensajes. De esta manera se evita que la aplicación APP quede bloqueada.

VI. CONCLUSIONES

BESA/ME provee un conjunto especial de componentes a nivel de la arquitectura para sistemas multiagentes ASMA que permite incluir en el sistema dispositivos móviles con soporte JME, principalmente teléfonos inteligentes y agendas digitales.

En el nivel agente, la solución se centra en proveer una clase especial de agentes que buscan proveer una alternativa a los problemas inherentes a los dispositivos móviles de limitada capacidad de procesamiento.

El agente "peer", es un tipo de agente especial que puede crearse en un entorno móvil o estático, haciendo posible crear agentes en dispositivos móviles y comunicarlos en dispositivos estáticos permitiendo en el desarrollo, dividir la funcionalidad entre estos.

En el nivel social, la solución se centra en proveer tres clases especiales de agentes que buscan proveer una alternativa a los problemas inherentes a los dispositivos móviles de limitada

capacidad de procesamiento, ayudándose de estos para cumplir con requerimientos funcionales.

El agente APS, es el único punto de entrada y salida de una aplicación, permitiendo solicitar y proveer servicios entre aplicaciones.

El agente AM, es un intermediario encargado de almacenar y recuperar los artefactos en un repositorio; el agente SSP, es un intermediario encargado de invocar servicios para acceder a diferentes recursos. Ambos permiten hacer que el proceso sea eficiente y de bajo acoplamiento.

Para superar la limitada capacidad de procesamiento, se provee la capacidad de interacción de agentes desde el entorno dinámico al estático y viceversa. Además de incluir de forma transparente la adición de nuevos contenedores en el entorno estático sin necesidad de reiniciar el sistema.

Para superar la poca capacidad de almacenamiento de los dispositivos móviles, se utilizan agentes intermediarios en el nivel social para acceder a la persistencia en el entorno estático.

Para superar la intermitencia de la comunicación, se utilizan componentes que verifican el estado de la comunicación. En caso que no exista comunicación, se almacenan temporalmente los mensajes localmente hasta que se restablezca. Además se utilizan mecanismos de tiempos límites para solicitar el reenvío de mensajes.

AGRADECIMIENTOS

A la Pontificia Universidad Javeriana por el apoyo a la Investigación. Este artículo hace parte de un proyecto de convocatoria interna sobre ampliar una plataforma de Sistemas MultiAgentes llamada BESA que incluya dispositivos móviles.

REFERENCIAS

[1] Bellifemine, F.; Caire, G.; Poggi, A. y Rimassa, G., 2003. JADE A White Paper, EXP Volumen 3, Número 3.

[2] Beneventano, D.; Bergamaschi, S.; Gelat, G.; Guerra, F. y Vinciniy, M., 2002. An agent framework for supporting the miks integration process. En: Proceedings of WOA.

[3] Cabri, G.; Leonardi, L. y Zambonelli, F., 2003. Brain: a framework for flexible role-based interactions in multiagent systems. En: Conference on Cooperative Information Systems.

[4] The Foundation for Intelligent Agents, 2009. FIPA Abstract Architecture Specification. En: IEEE Computer Society standards.

[5] González E.; Ávila J.A. y Bustacara C.J., 2003. BESA: Arquitectura para Construcción de Sistemas MultiAgentes. En: Conferencia Latinoaméricana de Informática, La Paz-Bolivia , pp. 70.

[6] González E.; Bustacara C.J. y Ávila J.A., 2003. Agents for Concurrent Programming. En: CPA 2003, Enschende-Holanda, IOS Press, pp 157-166.

[7] González E.; Avila J.A. y Bustacara C.J., 2003. BESA: Behavior-oriented, Event-Driven and Social-based Agent Framework. En: PDPTA'03, Las Vegas-USA, CSREA Press, vol. 3.

[8] Helsinger, A.; Kleinmann, K. y Brinn, M., 2004. A framework to control emergent survivability of multi agent systems. En: Proceedings of the AAMAS'04.

[9] Welch, P.; Hilderink, G.; Bakkers, A. y Stiles, G.,2001 . Safe and verifiable design of concurrent java programs. En: Proceedings of the IASTED International Conference. Software engineering and applications, pages 159 - 165, Scottdale - USA.

[10] SUN Microsystems, Inc., 2009. Java ME Technology. En: Community Development of Java Technology Specifications.

[11] SUN Microsystems, Inc.,2009. Java ME Technology -CLDC. En: Community Development of Java Technology Specifications.

[12] SUN Microsystems, Inc.,2009. The K Virtual Machine(KVM). En: Community Development of Java Technology Specifications.

[13] SUN Microsystems, Inc.,2009. Mobile Information Device Profile -JSR 118. En: Community Development of Java Technology Specifications.

[14] IEEE, 2009. IEEE 802.11x LAN/MAN Wireless. En: IEEE Standards Association.

[15] IEEE, 2009. IEEE 802.15 WPAN. En: IEEE Standards Association.

[16] Infrared Data Association, 2009. IrDA. En: IrDA Association.

[17] Botero, A.; Giraldo, H. y Moyano, A., 2004. MAD: MOBILITY SUPPORT BY AGENT-BASED DEVICES. Proyecto de Grado presentado para optar al título de Ingeniero de Sistemas Pontificia Universidad Javeriana, pages 44 - 93, Bogotá - Colombia.