TRABAJO ESPECIAL DE GRADO

ADAPTACIÓN DE PROTOCOLOS DE INTERACCIÓN DE SISTEMAS MULTIAGENTES A LA METAPLANIFICACIÓN DE PLATAFORMAS GRID

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Ing. Sumoza M., Rodolfo L.
Para optar al grado
de Especialista en Comunicaciones
y Redes de Comunicación de Datos

Caracas, 2007

TRABAJO ESPECIAL DE GRADO

ADAPTACIÓN DE PROTOCOLOS DE INTERACCIÓN DE SISTEMAS MULTIAGENTES A LA METAPLANIFICACIÓN DE PLATAFORMAS GRID

TUTOR ACADÉMICO: Prof. Jose Aguilar

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Ing. Sumoza M., Rodolfo L.
Para optar al grado
de Especialista en Comunicaciones
y Redes de Comunicación de Datos

Caracas, 2007

DEDICATORIA

A mi hija Sofía Gabriela y mi esposa Rosa. Ellas son todo para mi.

AGRADECIMIENTOS

Agradezco a mi Tutor el Dr. Jose Aguilar por toda su paciencia y colaboración en la elaboración de este trabajo.

A mi esposa Rosa por ayudarme a terminar el manuscrito del trabajo, y apoyarme en todo momento para su finalización.

A la Universidad Simón Bolívar por contribuir financieramente para la adquisición de los recursos que fueron necesarios.

Sumoza M., Rodolfo L.

ADAPTACIÓN DE PROTOCOLOS DE INTERACCIÓN DE SISTEMAS MULTIAGENTES A LA METAPLANIFICACIÓN DE PLATAFORMAS GRID

Tutor Académico: Prof. Jose Aguilar. Tesis. Caracas, U.C.V. Facultad de Ingeniería. Postgrado de Ingeniería Eléctrica. Año 2007, 94 p.

Palabras Claves: GRID, METAPLANIFICADOR, SISTEMAS MULTIAGENTES, PROTOCOLOS DE INTERACCIÓN.

Resumen. Los Sistemas Distribuidos Globales o GRIDs están orientados a dar repuesta a un gran conjunto de necesidades de ámbito computacional, que los sistemas tradicionales no han sido capaces de hacerlo. Siendo un campo de estudio relativamente nuevo presenta gran cantidad de temas que aun necesitan ser ampliamente desarrollados en el área de investigación, tal como el de la Metaplanificación. Este trabajo propone un Metaplanificador basado en Protocolos de Interacción de Sistemas Multiagentes, el cual se oriente a utilizar el paradigma de los modelos económicos como dinámica de interacción en la GRID.

INDICE

LISTA DE TABLAS Y FIGURAS

Pág.

LISTA DE TABLAS

LISTA DE FIGURAS

Caracas, Julio de 2007

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el ingeniero Rodolfo Sumoza, titulado:

"**Adaptación de Protocolos de Interacción de Sistemas Multiagentes a la Metaplanificación de Plataformas GRID**"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Grado de Especialista en Comunicaciones y Redes de Comunicación de Datos, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Jurado	Prof. Jurado

Prof. Jose Aguilar
Tutor

INTRODUCCIÓN

La evolución de la sociedad y su entorno ha llevado a los científicos del área de la computación a tener que adaptarse a nuevos requerimientos caracterizados por usuarios que cada día demandan más de las plataformas computacionales [2]. Inspirándose en ese hecho, la propuesta de este trabajo especial de grado se orienta a estudiar el tema de la computación global o ubicua, particularmente los ambientes GRIDs o mallas computacionales, los cuales hoy en día presentan un terreno fértil para la investigación. La computación GRID debe lograr dar soporte a un conjunto de servicios computacionales que impliquen el uso masivo de algunos componentes, como ejemplo se puede mencionar el manejo de grandes volúmenes de datos (almacenamiento de imágenes espaciales), o el de trabajar con cómputos científicos (predicción del estado del tiempo a escala mundial). En otras palabras, esta plataforma innovadora surge para dar respuesta a las limitaciones encontradas en las plataformas computacionales tradicionales, las cuales, dada la magnitud de las exigencias generadas desde distintos sectores científicos y tecnológicos, no pueden proveer la capacidad de cómputo o almacenamiento necesarios para satisfacer un problema dado, por lo que se deben incorporar todos los recursos disponibles en sitios dispersos geográficamente. Es ahí donde se pretende encontrar una salida al plantear las bondades de los ambientes colaborativos GRIDs, en los cuales se comparten recursos heterogéneos, geográficamente dispersos y dinámicamente disponibles [1, 4, 13, 14], con la intención de obtener una sinergia tecnológica ubicua.

En este sentido, son muchas las vertientes que se pueden pensar para desarrollar un tema de investigación, tales como el estudio de la seguridad, la planificación, entre otros. Actualmente se cuenta con los avances teóricos y prácticos, que tienen un cierto nivel de madurez en el área de los Sistemas Distribuidos tradicionales, lo que representa una ventaja ya que pueden ser aplicados en cierta medida en estas nuevas plataformas distribuidas. En este trabajo se estudia el problema de planificación a largo plazo, llamada también planificación global o metaplanificación, en plataformas

GRIDs. Específicamente, se desarrolló un protocolo basado Sistemas Multiagentes que se encarga de coordinar los procesos de la planificación a largo plazo. En el Capítulo I se describe el problema específico que se abordó, se plantean los objetivos del trabajo, su importancia y justificación. En el Capítulo II se presenta el marco referencial, que comprende una revisión de los estudios previos acerca del tema, los aspectos teóricos y conceptuales. En el Capítulo III se presenta la propuesta, donde se describen los métodos, técnicas y procedimientos utilizados. La propuesta del protocolo se presenta representándola de dos maneras, la primera es un macroalgoritmo que plantea paso a paso lo que hace el protocolo (se incluye una macroalgoritmo por cada uno de los enfoques planteados en la propuesta. Estos enfoques se basan en los posibles puntos de vista que se pueden tener en la GRID: el de los usuarios y el de los recursos). La segunda representación es gráfica, se le denomina Arquitectura de Agentes, y utiliza los modelos propuestos en AUML (Agent Unified Modeling Language). Específicamente, se utilizaron los diagramas de protocolos, con sus respectivas semánticas, uno para cada enfoque, y se complementaron con un diagrama de colaboración, junto con un diagrama de actividades. El Capítulo IV presenta las pruebas realizadas al protocolo propuesto, estas pruebas se basan en el modelado y la simulación, y específicamente se utilizó como herramienta de simulación el SimGrid, que es una librería escrita en lenguaje C. En este mismo capítulo se presenta la experimentación implementada en la simulación para generar los datos de prueba, y posteriormente se presenta el análisis de los resultados obtenidos.

CAPÍTULO I
PLANTEAMINENTO DEL PROBLEMA

Si se piensa en la gran cantidad de servicios demandados que sobre este tipo de plataformas pueden surgir, y que pueden provenir de cualquier parte del mundo con sólo hacer un click sobre algún portal GRID, se puede estar comenzando a entender la magnitud del compromiso que se tiene al tratar de dar respuesta a cada uno de ellos. Estas respuestas deben estar asociadas a procesos que involucren una estrategia de planificación muy rigurosa. La metaplanificación debe decidir si se aceptarán nuevas solicitudes introducidas en la plataforma, y debe garantizar la calidad del servicio a ofrecer para cada aplicación considerando la disponibilidad de recursos, maximizando el rendimiento esperado de la plataforma, optimizando los tiempos de respuesta, entre otros aspectos [2, 4, 9].

Este tipo de interacción tiene similitud con los procesos de oferta y demanda estudiados en las ciencias de economía y administración, por eso es interesante pensar en la idea que la planificación global debe tratar de encontrar la mejor combinación oferta y demanda de recursos, considerando algunas restricciones tales como: problemas en la red, intermitencia en la prestación de un servicio por parte de algún recurso, etc. Al tratar este tema se debe considerar la naturaleza de estas restricciones, las cuales pueden clasificarse en restricciones de tipo dinámico, como las ya mencionadas, y otro de tipo estático, como el poder o capacidad de cómputo de un determinado nodo, tema con el cual debe lidiar el planificador global al momento de decidir sobre la asignación de recursos.

La idea de base es aprovechar las bondades que ofrecen los Sistemas Multiagentes (SM), tales como su capacidad para resolver problemas de manera eficiente relacionados con los procesos de comunicación, coordinación e interacción entre los componentes de un sistema distribuido, así como también la posibilidad de darle autonomía a dichos componentes en los procesos de decisión, de tal forma de descentralizar esos procesos. Particularmente, se considera la utilización de los

protocolos de interacción de los SM utilizados en tareas de coordinación, para adaptarlos a las plataformas GRIDs. Estos protocolos están bien definidos a través de estándares establecidos por la organización FIPA (Foundation for Intelligent Physical Agents), y aprobados por la IEEE [12].

Esto apoya la propuesta de este trabajo al procurar adaptar estos protocolos a las plataformas GRIDs para la tarea de planificación a largo plazo. Así, este trabajo especial de grado propone la adaptación de un conjunto de protocolos de interacción basado en SM, particularmente los basados en el modelo de oferta y demanda, para realizar la planificación global en este tipo de plataformas. El protocolo resultante se modela y prueba sobre un ambiente de simulación para plataformas GRID, denominado SimGrid [20], elaborado principalmente para probar algoritmos de planificación en ambientes distribuidos.

CAPÍTULO II
MARCO REFERENCIAL

2.1. TRABAJOS RELACIONADOS

Una de las primeras propuestas que combina la teoría de agentes con la planificación en GRIDs, fue presentada en [9]. La idea de este trabajo es la de asignarle la ejecución de la planificación a los cliente. Para ello, cada cliente posee un agente, el cual tiene la responsabilidad de manejar las necesidades del usuario, entrando al mercado de recursos según vayan surgiendo las solicitudes, pero previamente este agente debe poseer la información de los recursos disponibles para poder tomar las decisiones de planificación, y poder dar después inicio al proceso de negociación. En otras palabras, requiere tener la información de los recursos disponibles y que se actualice constantemente. Además sólo plantea un enfoque donde los clientes deben competir por los recursos, y no en sentido inverso, en el que los recursos procuren competir para conseguir clientes, lo que implica que no considera gran parte de casos de negociaciones reales, como por ejemplo los que se pueden representar por medio de las licitaciones. La utilización de agentes en la planificación de GRIDs también se plantea en [17], sin embargo este trabajo es una propuesta que aun no se ha desarrollado ni implementado bajo los criterios que se plantean.

En [3] desarrollan un planificador a largo plazo utilizando como modelo económico el de las subastas Vickrey, y sólo utilizan el punto de vista de la competencia entre usuarios, y no entre recursos. En [4] hay una propuesta relacionada con la aplicación de los modelos económicos en la GRID, en este trabajo se describen varios modelos económicos que puede ser utilizados para el manejo de recursos y la planificación a largo y corto plazo. Entre estos modelos se mencionan los que se utilizan en este trabajo de grado: las licitaciones y las subastas. Para ello se considera que en la GRID los usuarios deben competir entre ellos por la asignación de recursos, y también los dueños de recursos deben competir entre si por la aceptación de los usuarios. Este trabajo no utiliza la teoría Multiagentes, y los procesos de interacción los ejecutan

aplicaciones distribuidas que se instalan en los distintos componentes GRID: portales, servidores, corredores, recursos.

En cuanto a lo planteado en [10], además de utilizar los modelos económicos para resolver el problema relacionado con la planificación a largo plazo en la GRID, lo simulan utilizando traces de cargas de trabajo reales, lo que les posibilitó probar su propuesta con mayor acercamiento a las verdaderas plataformas GRIDs. Al igual que antes, no usan la teoría de agentes en la propuesta. En [3] describen las características del proceso de planificación desde dos niveles: los planificadores a largo plazo, y los planificadores locales o a corto plazo.

Una evaluación de arquitecturas de metaplanificadores y políticas de asignación de tares para computación de alto rendimiento fue presentado en [6], en el cual modelaron y simularon un metaplanificador funcionando para una GRID constituida por varios clusters interconectados por una red WAN. Cada cluster internamente se interconecta a través de una red LAN, y su metaplanificador consiste en un conjunto de agentes funcionando en el lado de los recursos (como representantes de los recursos), los cuales tienen la tarea de verificar si el recurso está o no disponible. Se hay disponibilidad para el recurso, el agente le informa al coordinador central, que le envíe un tarea que se adecue a ese tipo de recurso, es decir, su filosofía es contraria a la de planificar en función de la demanda, y se enfoca en planificar en función de la oferta.

Existe otra gran cantidad de propuestas que pretenden resolver las dificultades presentes en los procesos de metaplanificación utilizando otros enfoques distintos a los modelos económicos y a la teoría de agentes, en [1, 10, 16, 24] se describen, se comentan y evalúan varias de estas alternativas, incluyendo las que actualmente son utilizadas en productos comerciales como CONDOR-G, GRaDS, NimrodG, UC4, etc. Además, mencionan los posibles enfoques que se pueden plantear para un planificador: centralizado, descentralizado o jerárquico.

2.2. PLANIFICACIÓN A LARGO PLAZO O METAPLANIFICACIÓN

En el ámbito de los ambientes GRIDs, los planificadores se clasifican según sus características y área de incidencia en dos, los de largo plazo, metaplanificadores o planificadores globales, y los planificadores locales o a corto plazo. En términos generales, un planificador a largo plazo puede verse como el ente encargado de descubrir recursos, de su selección y de la preasignación de trabajos a ellos, es importante indicar que la asignación en tiempo de uso de los recursos la realiza el planificador local o a corto plazo. El Metaplanificador interactúa con cierto tipo de servicio de información para buscar los recursos, hace la identificación dentro de la lista de los recursos disponibles, y con la debida autorización hace la asignación de los recursos considerando algún(os) criterio(s), además mantiene la información del estatus de los recursos. El proceso de selección de recursos es responsable por seleccionar aquellos que cumplen o satisfacen los requerimientos de trabajo solicitados [5].

La planificación a largo plazo, planificación global o metaplanificación, busca incrementar la eficiencia en la gestión de procesos que solicitan entrar en la GRID, al procurar disminuir los tiempos de respuesta, o al incrementar la tasa de respuestas satisfactorias, sin menoscabar la calidad del servicio ofrecido, tratando con los recursos provenientes de muchas y dispersas organizaciones que los ofrecen, y manejando las tareas generadas desde varios sitios. Como ya se dijo, incluye actividades como las de descubrir y preasignar recursos, calcular y estimar el rendimiento, entre otras cosas [5]. Este tipo de planificación es la que autoriza una nueva solicitud en la red pre-asignando los recursos, procurando que no se degrade el rendimiento del sistema, ya que de lo contrario la nueva solicitud debe esperar mientras puede ser atendida.

Por otro lado, los planificadores locales o a corto establecen las políticas específicas de asignación de un recurso para su uso. Tiene ingerencia directa sobre los procesos previamente preasignados por el metaplanificador de un entorno GRID, para poder

usar ese recurso. Este tipo de planificación utiliza los algoritmos clásicos de planificación de recursos, ya sea para entornos distribuidos o no, tales como planificadores de procesadores (round robin, FIFO, etc.), de discos (FIFO, SCAN, etc.), entre otros. La diferencia más notable entre el largo y el corto plazo es la de sus áreas de incidencia, la de largo plazo la tiene sobre el entorno global y ubicuo, y la de corto plazo la tiene sobre un recurso o grupos de recursos específicos.

2.3. SISTEMAS MULTI-AGENTES (SM)

En [15] se presentan algunas definiciones sobre el tema, inicialmente se plantea la definición de agente desde varias perspectivas, para luego plantear la definición asociada a las tecnologías de la información. En ese trabajo se define lo siguiente:

- Agentes:
  - Son entes que realizan una acción.
  - Son los que actúan en representación de otros (agente turístico, comercial, etc.).
- Agentes Software:
  - Son aplicaciones informáticas que tienen la capacidad de decidir cómo deben actuar para alcanzar sus objetivos.
  - Son sistemas informáticos que están situados en un cierto entorno y que tienen la capacidad de actuar autónomamente (sin la intervención de seres humanos u otros sistemas) para satisfacer los objetivos planteados en su diseño.
- Agentes Inteligentes:
  - Son agentes software que pueden funcionar confiablemente en entornos muy cambiantes e impredecibles.
- Sistemas Multiagentes:
  - Sistemas orientados a resolver problemas esencialmente distribuidos.
  - Sistemas basados en agentes y orientados a la resolución distribuida de problemas.

- Requieren que los problemas se subdividan en subproblemas, y a cada uno de estos se le asigna a un agente.
- Para resolver los problemas, los agentes pueden funcionar de manera coordinada, o individualmente.
- Este tipo de resolución de problemas es posible gracias a la capacidad de comunicación, cooperación y negociación que poseen los agentes.
- Cada agente del sistema tiene un punto de vista limitado (no tiene la información completa).
- No existe un control global para todo el sistema.
- Los datos están descentralizados.
- La computación es asíncrona.

En general, la teoría de agentes forma parte de un área de la Inteligencia Artificial llamada Inteligencia Artificial Distribuida (IAD), la cual procura desarrollar programas que puedan comunicarse entre sí a través de una red, y puedan establecer algunos tipos de negociaciones entre ellos, con el fin de resolver problemas con cierto nivel de complejidad que ninguno de ellos podría resolver por separado [15]. Así, son sistemas computacionales en los cuales varios agentes (programas) con cierto nivel de autonomía, interactúan entre sí para colaborar en la solución de un problema, o para intentar alcanzar una serie de objetivos individuales o colectivos. Estos agentes pueden ser de distintos tipos, es decir, diferentes entre ellos, y pueden o no tener metas comunes, pero siempre involucran comunicación entre ellos [15]. Entre sus características principales están:

- Capacidad de manejar conocimiento, lo que implica que puede conocer la información relacionada al estado de la plataforma (por ejemplo, la disponibilidad de sus recursos) y sus procesos, para tomar decisiones.
- Persecución del cumplimiento de objetivos. En este contexto, uno de ellos será el de mejorar el rendimiento de la plataforma.

- Capacidad de percibir eventos del entorno, el cual es un factor clave para interactuar con las actividades de oferta y demanda.
- Capacidad de influir en el entorno mediante acciones directas.

Algunas de las posibles características de un agente son:

- *Movilidad*: capacidad de transportarse de una máquina a otra.
- *Reacción*: actuación sobre el entorno mediante un comportamiento estímulo/respuesta.
- *Proacción*: toma la iniciativa para alcanzar sus objetivos.
- *Sociabilidad* o *cooperación*: capacidad de comunicarse con otros agentes, programas o personas.
- *Aprendizaje* o *adaptación*: comportamiento basado en la experiencia previa.
- *Continuidad temporal*: ejecución continua en el tiempo.
- *Carácter*: inclusión de estados de creencia, deseo e intención.

2.4. MODELOS DE MERCADO

En una estructura convencional del mercado, la comunidad de usuarios representa la demanda, y la comunidad de proveedores de recursos representa la oferta. En este sentido, dependiendo de la forma en que actúan y se relaciones estos dos grupos se pueden definir varios modelos económicos, tales como:

- Modelo de Mercado de productos: Los propietarios de recursos especifican el precio de sus servicios y le "facturan" a los clientes o usuarios de acuerdo a la cantidad de recurso que consumen. Las políticas de establecimientos de precios pueden ser únicas desde el inicio del proceso hasta su final (estáticas), o pueden variar durante el mismo. Pueden existir ciertas variaciones de este modelo tales como:

- Modelo de precios fijos: Es parecido al anterior, a diferencia que pueden existir ofertas especiales para atraer mayor cantidad de consumidores.
- Modelo de regateo o negociación: En este, los corredores (representantes de los clientes o de los usuarios) pueden intentar negociar o regatear para procurar lograr mejores precios con los proveedores.

- Modelo de Contratación y Oferta: Este modelo se basa en el hecho que los usuarios o clientes exponen o publican sus necesidades de recursos y servicios, y los dueños de productos chequean las características de estas solicitudes y pueden o no hacer contrataciones con los clientes que les convenga.
- Modelo de Subastas: Soporta las negociaciones del tipo "de uno a varios". Es decir, que varios clientes solicitan un mismo recurso, reduciendo la negociación a un solo valor (precio). Los subastadores configuran las reglas de la subasta, aceptables para los consumidores y los proveedores. Las subastas básicamente obligan a negociar un precio para el servicio. Los subastadores juegan el papel de mediador o coordinador, el cual anuncia los servicios y escucha las ofertas, los corredores, quienes son los representantes de los compradores, ofrecen una cantidad específica, pudiendo o no saber las cantidades ofrecidas por el resto de los corredores. El proceso podría terminar si el mínimo precio del servicio o recurso no es alcanzado en un determinado tiempo, o el subastador da el servicio o recurso a un ganador. Las subastas pueden realizarse de varias formas, entre las que se encuentran:
    - Subasta Inglesa: El subastador comienza con un precio bajo, y es incrementado por las distintas ofertas de los distintos clientes. Todas las ofertas pueden ser libremente incrementadas, superando otras ofertas. Cuando una oferta no es superada por ninguna otra, la subasta se termina, ganando la oferta más alta. Cada corredor decide cuanto ofertar. Cada ofertante conoce las ofertas de los demás.

- Subasta de única oferta, primer precio: Cada ofertante introduce su oferta, sin conocer las ofertas de los demás. La oferta más alta gana, y el recurso es dado con el precio de la mayor oferta.
- Subasta Vickrey: Es similar a la anterior, con la única diferencia, que el precio del recurso es establecido por la segunda oferta de mayor cantidad. En otras palabras, cuando se establece un ganador, el cual ofrece la mayor cantidad, este ganador deberá pagar la cantidad de la segunda oferta más alta.
- Subasta Alemana: El subastador comienza con un precio alto y lo va reduciendo hasta que algún cliente u ofertante esté dispuesto a pagar por ese precio.
- Subasta doble: Las órdenes de compra y las cotizaciones son introducidas en cualquier momento del proceso. Si en algún momento alguna orden de compra se equilibra o se iguala a una cotización (son compatibles), en términos de requerimientos y precios, la venta es ejecutada inmediatamente.

- Modelos de Licitaciones: En derecho público, acto por el cual el Estado concede contratos para la ejecución de obras de interés público; la adjudicación definitiva de dichos contrato es precedida de un concurso o subasta en el que varias personas, naturales o morales, presentan sus cotizaciones o los precios que cobrarían por la ejecución del contrato [21].
- Modelos de Trueques: Intercambio de objetos o servicios por otros objetos o servicios, se diferencia de la compra/venta en que no aparece dinero [23].

2.4.1. Modelos Económicos para la GRID

En [4] se plantea la utilización del paradigma económico para introducirlo como metodología en el manejo de recursos en la GRID. En ese trabajo se expone que el enfoque económico provee una base sólida para el manejo exitoso de la descentralización y heterogeneidad presentes en las economías humanas. Los modelos económicos basados en la oferta y demanda pueden proveer algoritmos,

políticas y herramientas para localizar y compartir recursos en la GRID. Estos modelos pueden estar basados en el trueque o en la asignación de precios. En el modelo de trueque todos los participantes necesitan poseer sus propios recursos para poder realizar los intercambios. Por otro lado, el modelo basado en precios, a los recursos se les asigna un precio basado en la demanda, la oferta, el valor, y el estado del sistema económico. En ese trabajo sólo se limitan a comentar en ámbitos generales la utilización de este enfoque en las plataformas GRIDs.

Dada la naturaleza distribuida y heterogénea de la GRID y su manera de funcionar, se puede hacer una analogía con el mercado humano visto desde una perspectiva amplia (global). En este mercado el sistema de manejos de recursos necesita proveer mecanismos y herramientas que sean útiles al momento de conseguir las metas establecidas por los propietarios de recursos o proveedores, y por los usuarios o clientes. Los consumidores de recursos (clientes) necesitan un modelo basado en la utilidad, que les permita satisfacer específicamente los requerimientos que tengan bajo sus parámetros de preferencia, y a través de los "corredores" (tal como en el mundo económico real) pueden proveer estrategias para la escogencia apropiada de recursos que cumplan con sus requerimientos. Por otro lado, los propietarios de recursos junto con los recursos que están dispuestos a ofrecer necesitan mecanismos para establecer esquemas de generación de precios que incrementen la utilización de sus sistemas y herramientas.

En este sentido, en el mundo GRID, se pueden establecer dos grupos, el que demanda recursos y servicios y el que posee dichos recursos y presta servicios. El primer grupo puede estar conformado por usuarios finales, aplicaciones, dispositivos, redes o nodos de la GRID completos, etc. El segundo grupo lo representan todos aquellos propietarios de recursos que ofrecen sus servicios en la GRID. Tal como en el mundo de la economía humana, en la GRID, la naturaleza

de estos grupos es completamente dinámica, es decir, posee una naturaleza cambiante y casi impredecible.

Nombraremos en especifico el trabajo de [4], ya que es el trabajo que sirve de base al nuestro. En ese trabajo se menciona que un modelo de mercado aplicado a la GRID está compuesto por "participantes" que se pueden clasificar en varios tipos, cada uno relacionado con los grupos antes mencionados: *los proveedores de servicios* (dueños de recursos), que juegan el rol de los productores, *los corredores* que representan a los consumidores, es decir, no son los consumidores como tal, y *los consumidores* que interactúan con sus propios corredores para manejar la planificación de sus requerimientos en la GRID. Cada consumidor puede tener un corredor particular y pueden existir corredores que atiendan a varios consumidores. La idea de la existencia de los corredores es crear un representante virtual de los clientes. Existe un cuarto tipo de jugador que es el *Manejador de Servicios*, el cual es un sistema manejado por los proveedores para ofrecer los servicios de los recursos que estén dispuestos a vender, según el tipo de configuración un manejador puede representar a uno o a varios recursos, que pueden o no ser del mismo tipo. La interacción entre los corredores y los manejadores de servicios durante las negociaciones comerciales está controlada por un último tipo de jugador: *el metaplanificador*, el cual está compuesto por elementos que colaboran entre sí para llevar a cabo las funciones antes descritas, estos elementos son: el portal web o la interfaz hacia el cliente, que le permite a los usuarios y dueños de recursos interactuar en la GRID, el buscador general, encargado de la búsqueda inicial basada en requerimientos previamente establecidos (antes de iniciar la búsqueda); el preseleccionador, el cual utiliza los recursos encontrados por el buscador, y los selecciona según determinadas estrategias o criterios de escogencia. Para esto ultimo requiere de elementos que se encarguen de negociar las contrataciones entre los clientes y los dueños de recursos, permitiendo de esta forma la asignación final de los recursos, utilizando para ello esquemas como las licitaciones o subastas, entre otros, que permitan

llegar a un mutuo acuerdo entre los entes involucrados. Así, estos últimos pasos deben ser ejecutados por otros componentes del planificador que se encarguen de las tareas de interacción cliente-recurso, y sus respectivas contrataciones.

Como se dijo antes, estos entes pueden utilizar varios modelos económicos o protocolos de interacción para tomar decisiones. En este contexto, los usuarios, los dueños de recursos, y los recursos mismos son entes físicos o "reales", y participan en la GRID a través de sus representantes virtuales o digitales, corredores y manejadores de servicios, respectivamente. Ese trabajo es el insumo inicial de nuestra propuesta.

2.5. SIMULACIÓN PARA LA GRID

Una de las alternativas que existe para probar los sistemas consiste en realizar la fase de verificación sobre un ambiente real, pero pueden presentarse casos en los que las condiciones de los ambiente reales no son las más adecuadas para obtener la información pertinente en cuanto al establecimiento de la validez del diseño planteado. Esto se debe, entre otras cosas, a que en los escenarios reales existen muchos factores o variables que no son controlados por el experimentador, y que al necesitar, por ejemplo, generar repeticiones de las pruebas, las condiciones en cada repetición varían, perdiendo la validez de los ensayos. También, las verificaciones en ambientes reales son más difíciles de manejar porque los sistemas en funcionamiento o producción disminuyen la capacidad de maniobra del experimentador. Todas esas dificultades se pueden superar creando un modelo del diseño planteado, y sometiéndolo a pruebas en ambientes controlados bajo los esquemas de las simulaciones, donde cada una de las variables involucradas son conocidas, dándole el control total al experimentador.

La fase de pruebas de un protocolo para la metaplanificación en ambientes GRID, tal como en otros sistemas, posee las características planteadas en el párrafo anterior, y es por ese motivo que para este trabajo se realizaron las pruebas a través de un

modelo simulado por medio de un conjunto de librerías y funciones implementados en el lenguaje de programación C, denominado SimGrid [7, 8], que es una herramienta de simulación para el estudio de algoritmos de planificación en aplicaciones distribuidas, y su objetivo principal es el de dar apoyo a la comunidad orientada a la investigación en esta área. SimGrid provee un conjunto de funcionalidades que permiten construir simuladores para aplicaciones y topologías de ambientes computacionales específicas. Se basa en el desarrollo de simulaciones manejadas por eventos, fundamentadas en el modelado de recursos, dándole a cada uno dos componentes descriptivos básicos: la latencia (tiempo en segundos para acceder a un recurso), y la tasa de servicio (número de unidades de trabajo ejecutadas por unidad de tiempo). Estos componentes son modelados a través de vectores de valores de latencias o tasas de servicios a través del tiempo, denominados trazas. El modelado de las interconexiones en la topología de red es independiente al de los recursos, permitiendo establecer varias vías posibles de conexión entre los diferentes componentes, lo cual permite modelar las posibles rutas de transmisión establecidas en ambientes de red donde se involucran los enrutadores. Cada enlace se describe a través de su capacidad de transmisión y su latencia asociada.

La transferencia de datos y los procesos de cómputo son vistos ambos como ejecución de tareas. Siendo responsabilidad del programador (usuario de la herramienta), asegurar que las tareas asociadas a cómputos sean planificados sobre los procesadores, y las tareas asociadas a las transferencias de datos se ejecuten sobre los enlaces de red. SimGrid manipula dos tipos de estructuras de datos: una para el manejo de recursos y otra para el manejo de tareas. Los recursos son descritos por su nombre, por un conjunto de métricas, trazas o valores constantes, por ejemplo, un procesador se describe por una medida de su velocidad y una traza de su disponibilidad. Un enlace se describe por una traza de su latencia y una traza de su ancho de banda disponible. Una tarea es descrita por un nombre, un costo, y un estado (planificado, no-planificado, en ejecución, listo y completado). Para el costo se utiliza en el caso de la transferencia de datos la cantidad de bytes utilizadas por

unidad de tiempo (en segundos), y para el caso de los cómputos se utiliza la cantidad de tiempo en segundos requerida en la ejecución del proceso que se esté referenciando.

En resumen, Simgrid permite hacer simulaciones con modelos de redes mucho más reales, y trabajar con niveles de complejidad mayores al permitir mayores niveles de abstracción. En relación a los Sistemas Multiagentes, Simgrid, en su última versión (v.2) incorpora la capacidad de trabajar con agentes planificadores distribuidos, es decir, cada recurso, con su código asociado, representa un agente dentro de la plataforma distribuida.

Simgrid utiliza un conjunto de valores enmarcados en el tiempo denominados trazas, que son los valores que describen a cada uno de los componentes involucrados en la GRID, tanto a nivel local para cada nodo (capacidad de CPU), como a nivel de interconexiones (anchos de banda y latencia), con las cuales se representan las plataformas y topologías bajo estudio, según los datos que se tengan de cada escenario en particular. Estas trazas son utilizadas como insumo para generar las distintas ejecuciones de las simulaciones, sin embargo, cada caso a simular puede presentar características y necesidades muy distintas entre si, por lo cual esta herramienta ofrece un conjunto de módulos para la simulación de distintos tipos de escenarios. Por ejemplo, el módulo SimDag, se utiliza para representar ambientes que han sido descritos a través de grafos acíclicos dirigidos, o el módulo SMPI, se utiliza para simular aplicaciones paralelas ejecutadas en MPI. Para este trabajo en particular se utilizó el módulo MetaSimgrid o MSG, el cual permite hacer simulaciones de índole general de aplicaciones distribuidas.

CAPÍTULO III
PROPUESTA

3.1. PROTOCOLO METAPLANIFICADOR PROPUESTO

La propuesta de este trabajo es diseñar un Planificador a Largo Plazo en Plataformas GRIDs. Antes de explicar con detalle la adaptación de los protocolos de interacción de SM a la Planificación a Largo Plazo en Plataformas GRIDs, es necesario explicar el entorno que se utiliza como marco de referencia, detallando el listado de participantes, y el rol que cada uno de ellos tiene en la GRID. En la figura 1 se pueden apreciar dichos participantes y su interacción, esta figura es una representación lógica, es decir, es una representación simbólica que esquematiza la forma en que están relacionados los involucrados, según los jugadores antes mencionados: clientes y dueños de recursos.

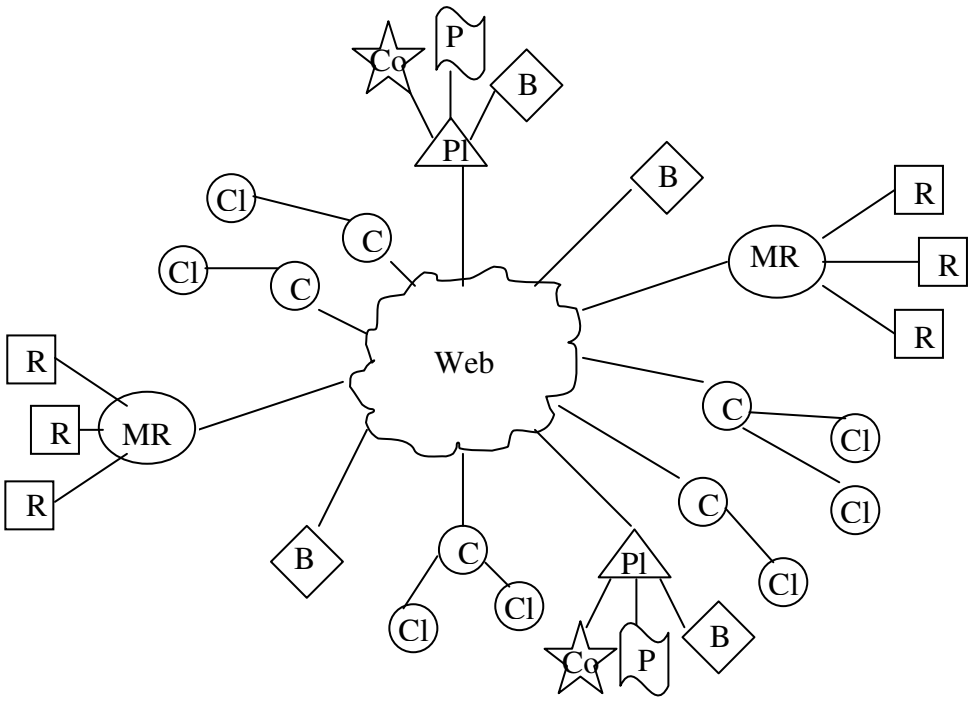


Figura 1. Esquema General clásico de la GRID

Donde:

- R: Dueño de Recursos y sus recursos ofrecidos, los cuales son los proveedores de servicios, constituyen un conjunto de recursos ofrecidos en la GRID. Estos recursos pueden estar en distintos sitios geográficos en la red, y pueden establecer grupos o asociaciones para satisfacer un requerimiento en un determinado momento. Este participante tiene un perfil real o físico.
- MR: Manejador de Recursos o Servicios. Son sistemas que trabajan para controlar las agrupaciones de recursos, es decir, es el representante digital o virtual de los recursos, y por implicación directa, de los dueños de los recursos. Tienen el rol de informar los términos bajo las cuales se realizarán las propuestas de servicios. Este participante es el que funge como proveedor de servicios en la GRID.
- Cl: Clientes o Usuarios, pueden ser dispositivos, aplicaciones, nodos, otros recursos, etc., lo cuales están interesados en utilizar un determinado recurso. Son participantes físicos o reales.
- C: Son los corredores, que al ser participantes virtuales o digitales, representan a los clientes o usuarios en la GRID.
- Pl: Metaplanificador, es un participante virtual que controla o coordina el proceso de planificación a largo plazo. Tiene que mediar entre los corredores y los manejadores de recursos. Es el ente regulador de la interacción entre los clientes y los recursos (dueños de recursos), y es el encargado de la planificación global como tal. Está compuesto por varios elementos, tales como los buscadores (pueden ser uno o varios, o utilizar buscadores independientes, o de otros portales, etc.), los portales, y los elementos de asignación de recursos (coordinación, selección, contratación, etc.).
- B: Buscadores de Recursos. Son aplicaciones que forman parte de los metaplanificadores. Existen algunas herramientas que realizan este tipo de labor, y se utilizarán para completar el planificador propuesto.

Su rol es el de buscar los recursos disponibles en la GRID que cumplan con los criterios suministrados por el cliente. Los buscadores pueden ser parte integral de los metaplanificadores, o pueden ser buscadores independientes. Lo estrictamente necesario es que todo planificador debe tener asociado al menos un buscador.

- Co: Coordinador, se encarga de coordinar la interacción (asignación de recursos), son los encargados de realizar las tareas de selección y contratación, incluyendo el establecimiento de los acuerdos entre los involucrados.
- P: Portales, es la interfaz (preferiblemente Web), entre usuarios y recursos, y la puerta de entrada de los usuarios a la GRID.

En el esquema anterior se puede apreciar que se está utilizando una clasificación para los involucrados o participantes en la GRID, la cual permite dividirlos en dos grupos: los participantes reales o físicos, y los participantes virtuales o digitales. Con estos últimos son los que se trabaja directamente en el metaplanificador propuesto.

3.1.1. Caracterización del Proceso de metaplanificación

- Cuando un cliente solicita un servicio, lo hace a través de uno de los portales que están distribuidos geográficamente y conectados vía Web, y son parte constituyentes de los Planificadores. En los servidores planificadores (Pl) se encuentran los demás elementos del metaplanificador. Al interactuar con uno de estos, el componente Pl le entrega una instancia del agente "corredor". Una vez que el planificador le asigna una instancia del agente corredor, ésta comienza a dialogar con el usuario o cliente para levantar la información sobre los recursos que necesita. Una vez hecho esto, esta información es enviada al planificador para que éste se encargue de asignarle la tarea al buscador (pueden ser varios buscadores al mismo momento) para que inicie el proceso de búsqueda. Cuando se ha mencionado que un usuario necesita de un recurso, se

puede estar refiriendo en realidad a un conjunto de recursos que de forma grupal logran satisfacer su requerimiento.

- Los Planificadores interactúan con los buscadores (cada Planificador puede poseer varios buscadores o utilizar buscadores de otros portales o independientes), con los agentes Manejadores de Recursos, y con los Corredores. Cada Planificador puede interactuar con varios buscadores al mismo tiempo, para cada solicitud.
- Los Manejadores de Recursos son los encargados de publicar, con todas las características y restricciones, los recursos ofrecidos por los propietarios. Puede existir un manejador para varios propietarios de recursos.
- Los buscadores tienen la labor de la localización de recursos que cumplan con los requerimientos de los usuarios, tales como sus características estáticas, por ejemplo la velocidad de un procesador.
- Los usuarios o clientes están representados por corredores, estos corredores tratan de conseguir el mejor recurso para su cliente. Un corredor puede darle soporte a varios clientes.
- Cuando un dueño de recurso (pueden ser varios dueños agrupados) quiere ofrecer sus servicios, lo debe hacer a través de algún manejador de recursos, ya sea propio o compartido.
- Cada cliente puede tener asociado uno o varios procesos que se ejecutan para llevar a cabo su solicitud. Cada proceso se representa como una secuencia de tareas donde cada una de ellas requiere distintos tipos de recursos computacionales para ser completadas. Este cliente puede estar representado por aplicaciones, usuarios finales, dispositivos, etc.
- El proceso de selección final de recursos, contratación y asignación, es realizado por los coordinadores, encargados de la interacción del Planificador. Estos componentes se encargan de la preasignación o autorización de uso del recurso, luego de ésta, le otorgan la responsabilidad de la ejecución de la tarea a los planificadores locales.
- En la estructura multiagentes, los MR, B, C y Co son los agentes.

- El coordinador (Co), es el agente que coordina las negociaciones entre los participantes, y forma parte del Pl.

Dada las similitudes que existen entre los tipos de interacciones (negociaciones) de los participantes en la GRID y los procesos económicos humanos, dicha interacción puede ser guiada por uno o varios de los modelos económicos antes mencionados. En la actualidad, algunos de estos modelos han sido descritos a través protocolos de interacción por FIPA. En este trabajo se utilizan algunos de estos protocolos como base para el diseño propuesto, y la planificación se plantea a través de dos enfoques, que al combinarse generan un enfoque híbrido, el cual representa el esquema final implementado para la fase de simulación. El primer enfoque se establece desde el punto de vista de los usuarios o clientes, denominado en este trabajo como el *enfoque-cliente*, donde los manejadores de recursos a través del planificador intentan conseguir el interés de adquisición de los clientes, este proceso se representa por la estructura de licitaciones (modelo económico) donde existen varios vendedores y un comprador. Los intereses que son atendidos son los de los clientes, de esta forma el objetivo principal es que éstos puedan adquirir los recursos con las mejores prestaciones y al mejor precio. El otro enfoque es desde el punto de vista de los recursos (realmente es desde el punto de vista de los dueños de los recursos), denominado *enfoque-recurso* en este trabajo, donde la idea de base es conseguir que el cliente proponga su mejor oferta, según las especificaciones del vendedor. Este proceso es similar al de las subastas (otro modelo económico), donde existen varios compradores interesados y un solo vendedor. En este caso, los intereses que son considerados son los de los dueños de recursos, y su objetivo principal es conseguir vender los recursos o servicios al cliente que ofrezca las mejores condiciones.

3.1.1.1. Enfoque Cliente

En el enfoque-cliente, el proceso se inicia cuando los usuarios o clientes manifiestan a través del corredor su interés en adquirir un recurso (o grupos de recursos), esta acción activa al buscador o buscadores asociados al planificador.

Como se mencionó antes, el buscador forma parte del planificador, pero en este trabajo no se incluye la elaboración de buscadores ya que existen herramientas que hacen esta labor. Algo importante a resaltar es que pueden ser utilizados varios buscadores en una misma solicitud. Cuando el buscador o los buscadores consiguen el listado de recursos disponibles, el planificador inicia la labor de la preselección de los recursos a través del proceso de interacción.

Es importante aclarar que la necesidad que especifica cada cliente puede involucrar a varios tipos de recursos, así el buscador debe conseguir una lista de recursos disponibles por cada tipo de recurso requerido. Una vez que se seleccionan todos los recursos necesarios, se ejecuta la contratación con los dueños. Esta contratación se debe realizar al final de la preselección, y no al momento en que se preselecciona cada recurso, porque puede existir la posibilidad que algún recurso requerido no pueda ser contratado, y por ende el cliente no puede satisfacer su demanda ya que necesita al grupo de recursos en su totalidad. En otras palabras, el acuerdo final para la contratación se hace a través del establecimiento de un mutuo acuerdo entre los clientes y los dueños de recursos, el cual se hace al final del proceso de preselección, donde se desarrollan las actividades de interacción adaptadas a un modelo económico tal como el de las licitaciones. Así, cada vez que se selecciona un recurso se hace a través de un proceso de licitación, el cual involucra todo el listado de requerimientos que introdujo el cliente, es decir, se establece lo que se denomina un pliego licitatorio, el cual contiene toda la información necesaria para realizar la contratación. De esta forma, lo que se plantea en este trabajo es la adaptación de un grupo de protocolos de interacción de sistemas multiagentes de FIPA, basándose en el modelo de licitaciones, para llevar a cabo este proceso en particular. Para describir completamente este enfoque se presenta a continuación el macroalgoritmo del mismo:

- Macroalgoritmo:
    A. Ejecutar los siguientes pasos, al momento en que un cliente genera una solicitud de un recurso
        a. Invocar el servicio del portal, alojado en el servidor planificador correspondiente para entrar en la GRID. Dado que existen varios tipos de clientes: usuarios finales, aplicaciones, etc., el servicio de portal interactuará de distintas formas según sea el tipo de cliente. Por ejemplo, si es un cliente final, la interacción se realiza a través de una página web; por otro lado, si es una aplicación está debe interactuar a través de negociaciones directas con el servidor planificador, sin la asistencia de interfaces web.
        b. Por razones de seguridad, una vez conectado con el portal se debe ejecutar un proceso de identificación y autenticación, sea cual fuese el tipo de cliente.
        c. Introducir, en la opción referente a búsqueda y asignación de recursos, el listado o compendio de requerimientos.
        d. El servidor planificador asigna y activa un corredor, que es la aplicación (agente) que atenderá tal requerimiento del cliente.
        e. Una vez activado el corredor, éste interactúa con el planificador (buscador, coordinador), el cual internamente inicia el proceso de metaplanificación.
        f. El corredor le envía el listado de requerimientos al servidor de planificación.
    B. Llevar a cabo los siguientes procesos dentro de la metaplanificación:

a. Dado el listado de requerimientos enviados por el corredor, el metaplanificador, alojado en el servidor Pl, activa al buscador general.
b. El buscador inicia el proceso de búsqueda con los requerimientos suministrados.
c. El buscador genera una lista de recursos. Esta lista agrupa los recursos encontrados según su tipo, es decir, muestra una lista de recursos por cada tipo.
d. Una vez que se tenga la lista de recursos encontrados, se activa el proceso de selección a través de la utilización del protocolo de interacción tipo licitación. El agente coordinador interactúa con los manejadores de servicios para seleccionar un recurso por tipo, buscando maximizar los beneficios del cliente. El proceso de selección se realiza utilizando el listado de requerimientos completo suministrado por el usuario, y los criterios orientados a satisfacer los parámetros de calidad de servicio.
e. Este protocolo de interacción utiliza los requerimientos del usuario como pliego de licitación. Este pliego es el patrón que sigue para buscar el mejor recurso entre la lista, para lo cual se utilizan algunos indicadores de rendimiento: características de los enlaces de red, características estáticas del recurso, etc.
f. El proceso de interacción, que para este enfoque es la licitación, consiste en procurar establecer acuerdos con los manejadores de recursos para lograr minimizar los costos del mismo previa revisión del cumplimiento del pliego de licitación establecido. Al ser un proceso de licitación, los manejadores de servicios encontrados en

la búsqueda inicial ofrecen su mejor propuesta para intentar ganar la buena pro del cliente.

g. La interacción (licitación) finaliza cuando se ha encontrado un recurso que cumple con lo establecido en el pliego, y éste ofrece el menor costo de contratación.
h. Una vez finalizada la selección de todos los recursos necesarios para cumplir con los requerimientos del cliente, se establece un acuerdo con todos los manejadores de los recursos seleccionados. Este acuerdo consiste en indicarle a cada extremo de la negociación (cliente y recursos) sobre la asignación del recurso, Una vez establecido dicho acuerdo, con todos los recursos necesitados, se realiza la contratación con todos ellos.
i. Después de realizar dicha contratación, se da la autorización al cliente para que utilice los recursos según lo acordado: precios, tiempo de uso, restricciones. Esto implica decirle al recurso la tarea que se le asignó. Dicho recurso, según sus mecanismos de planificación, gestionará el uso del recurso por parte del usuario. Esto lo desarrolla a través de su planificador a corto plazo, que actúa a partir de ese momento.

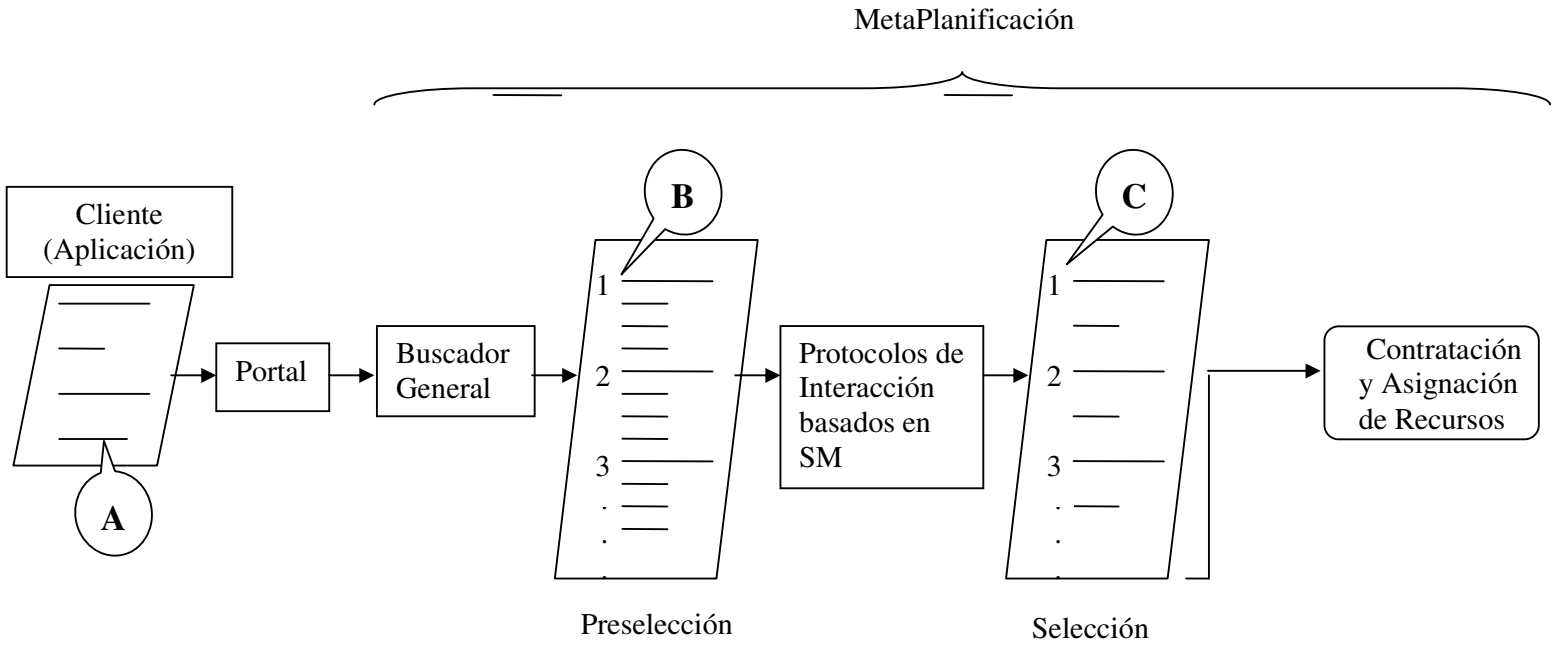
Según lo anterior, se puede lograr cumplir de forma general con los requerimientos, buscando maximizar los beneficios del cliente, esto puede verse concretamente en la figura 2, donde se sintetiza el proceso planteado para este enfoque, comenzando en la parte izquierda, el proceso se inicia cuando el cliente (por ejemplo una aplicación), le envía al servidor planificador el listado que contiene los detalles de su requerimiento, para conformar lo que se ha

denominado el pliego de licitación. El servidor activa al buscador general, el cual suministra un listado de grupos (lista de listas) según los tipos de recursos necesitados. Este listado de grupos se aprecia en **B**, donde cada grupo: 1, 2, 3… es una lista de potenciales recursos según su tipo. Con este listado, el protocolo de interacción realiza el proceso de selección (seleccionan un recurso por tipo usando el esquema de licitación), según el perfil buscado y la minimización de los costos asociados. Este listado se puede observar en C, donde 1, 2, 3… son los recursos seleccionados por tipo (uno por tipo). Al finalizar la selección de los recursos necesarios, se genera el acuerdo de extremo a extremo. Finalmente se realiza la contratación y asignación de recursos.

3.1.1.2. Enfoque Recurso

En el enfoque-recurso la idea es que cuando existan varios clientes interesados en utilizar los servicios de un recurso, se puede iniciar un proceso donde se hace la contratación con el cliente que ofrezca las mejores condiciones de contratación. Este tipo de proceso es similar al de las subastas, donde existen varios clientes interesados en un solo recurso ofrecido (varios compradores y un vendedor). En este caso, cuando cada cliente ha preseleccionado un recurso, lo cual lo puede hacer a través de un proceso aleatorio de escogencia o escogiendo el primero de la lista que genera el buscador general, el cliente entra en el proceso de la subasta para intentar ganar el recurso ante el resto de los clientes que también lo seleccionaron.

Para describir completamente este enfoque, se presenta a continuación el macroalgoritmo del mismo:



A= Requerimientos del Cliente.
B= Lista de recursos encontrados agrupados por el tipo requerido.
C= Lista de recursos seleccionados por tipo requerido.

Figura 2: Enfoque Cliente de la Metaplanificación

- Macroalgoritmo:
    A. Ejecutar los siguientes pasos, al momento en que un cliente genera una solicitud de un recurso:
        a. Invocar el servicio del portal, alojado en el servidor planificador correspondiente para entrar en la GRID. Dado que existen varios tipos de clientes: usuarios finales, aplicaciones, etc., el servicio de portal interactuará de distintas formas según sea el tipo de cliente. Por ejemplo, si es un cliente final, la interacción se realiza a través de una página web, por otro lado si es una aplicación, está debe interactuar a través de negociaciones directas con el servidor planificador, sin la asistencia de interfaces web.
        b. Por razones de seguridad, una vez conectado con el portal, se debe ejecutar un proceso de identificación y autenticación, sea cual fuese el tipo de cliente.
        c. Introducir, en la opción referente a búsqueda y asignación de recursos, el listado o compendio de requerimientos.
        d. El servidor planificador asigna y activa un corredor, que es la aplicación que atenderá tal requerimiento por parte del cliente.
        e. Una vez activado el corredor, éste interactúa con el servidor planificador, el cual internamente inicia el proceso de metaplanificación.
        f. El corredor le envía el listado de requerimientos al servidor.
    B. Llevar a cabo los siguientes procesos dentro de la metaplanificación:
        a. Dado el listado de requerimientos enviados por el corredor, el metaplanificador, alojado en el servidor Pl, activa el buscador general.

b. El buscador inicia el proceso de búsqueda con los requerimientos suministrados.
c. El buscador muestra la lista de grupos de recursos sugeridos que satisfacen los requerimientos de los clientes.
d. Una vez que se tenga el recurso seleccionado, se activa el proceso de selección de un cliente, dentro de un conjunto de clientes interesados y que han ejecutado también los pasos previos, a través de la utilización del protocolo de interacción tipo subasta. El agente coordinador interactúa con los corredores para seleccionar un cliente, buscando maximizar los beneficios del dueño del recurso. El proceso de selección se realiza utilizando el listado de requerimientos completo suministrado por el dueño del recurso en cuanto a lo que debe satisfacer el cliente.
e. En Pl se genera un perfil con los requerimientos del dueño del recurso. Este perfil es el patrón que sigue el coordinador para buscar el mejor cliente entre la lista. La lista de clientes interesados contiene la información asociada a indicadores de confianza como: contrataciones anteriores, solvencia del cliente, y características del cliente en particular.
f. La interacción consiste en procurar establecer entre el grupo de los cliente interesados al ganador, a través de un proceso de subasta, buscando maximizar los beneficios del dueño del recurso. La interacción finaliza cuando se ha encontrado un cliente que cumple con el perfil buscado por el dueño del recurso y ofrezca el mayor monto.
g. Una vez finalizada la selección del cliente, se realiza el acuerdo con el manejador del recurso. Este acuerdo consiste en indicarle a cada extremo de la

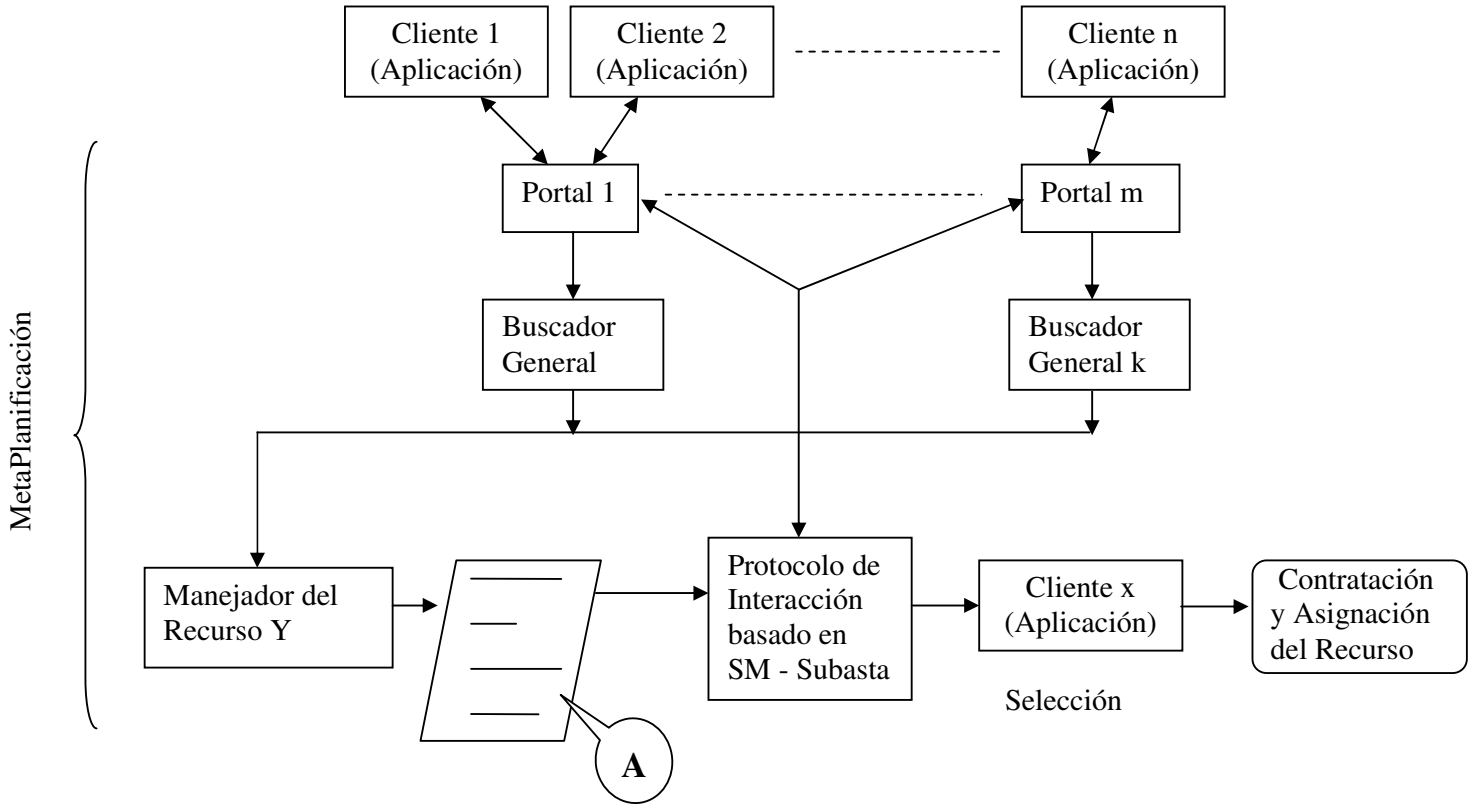
negociación (cliente y recurso) la asignación. Una vez hecho el acuerdo, se realiza la contratación entre el cliente y el dueño del recurso.

h. Después de realizar dicha contratación, se da la autorización al cliente para que utilice el recurso según lo acordado: precio, tiempo de uso, restricciones.

Según lo anterior, se puede lograr cumplir de forma general con los requerimientos, buscando maximizar los beneficios del dueño del recurso, esto puede verse concretamente en la figura 3. En esta figura se sintetiza el proceso planteado para este enfoque, comenzando en la parte superior, el proceso se inicia cuando los clientes (por ejemplo las aplicaciones), les envían a través de los portales el listado que contienen los detalles de sus requerimientos. El servidor planificador activa a uno o varios buscadores, los cuales suministran la información sobre el recurso necesitado y encontrado. Con esto, cada recurso, a través de su manejador, activa el protocolo de subasta para realizar el proceso de selección de los clientes. Finalmente se realiza la contratación del recurso.

3.1.1.3. Enfoque Híbrido

En los dos enfoques anteriores se plantearon algunas de las situaciones probables que se pueden presentar en el momento de establecer un proceso de "negociación" sobre algún servicio o producto, se presentaron escenarios donde pueden concurrir varios ofertantes de productos o servicios vs. un solo comprador o cliente, y el escenario inverso donde existen varios clientes para un solo recurso, pero es interesante plantearse interrogantes como ¿qué ocurre si en el enfoque cliente al tener el listado de recursos (lista de listas), uno de esos recursos está siendo requerido por otros clientes o sólo existe un vendedor?, ¿qué ocurre cuando sólo existen un solo comprador y/o un solo vendedor?. Para dar respuesta a este tipo de interrogantes, que le dan un matiz mucho más real a los escenarios planteados, surge la idea del enfoque hibrido, donde se combinan las propuestas del enfoque cliente y el enfoque recurso, según sea el caso.



A= Requerimientos del Dueño del Recurso Y.

Figura 3: Enfoque Recurso de la Metaplanificación

- Macroalgoritmo:
    A. Ejecutar los siguientes pasos, al momento en que un cliente genera una solicitud de un recurso
        a. Invocar el servicio del portal, alojado en el servidor planificador correspondiente para entrar en la GRID. Dado que existen varios tipos de clientes: usuarios finales, aplicaciones, etc., el servicio de portal interactuará de distintas formas según sea el tipo de cliente. Por ejemplo, si es un cliente final, la interacción se realiza a través de una página web, por otro lado si es una aplicación, está debe interactuar a través de negociaciones directas con el servidor planificador, sin la asistencia de interfaces web.
        b. Por razones de seguridad, una vez conectado con el portal, se debe ejecutar un proceso de identificación y autenticación, sea cual fuese el tipo de cliente.
        c. Introducir, en la opción referente a búsqueda y asignación de recursos, el listado o compendio de requerimientos.
        d. El servidor planificador asigna y activa un corredor, que es la aplicación (agente) que atenderá tal requerimiento del cliente.
        e. Una vez activado el corredor, éste interactúa con el planificador (buscador, coordinador), el cual internamente inicia el proceso de metaplanificación.
        f. El corredor le envía el listado de requerimientos al servidor de planificación.

    B. Llevar a cabo los siguientes procesos dentro de la metaplanificación:
        a. Dado el listado de requerimientos enviados por el corredor, el metaplanificador, alojado en el servidor Pl, activa el buscador general.
        b. El buscador inicia el proceso de búsqueda con los requerimientos suministrados.

c. El buscador genera una lista de recursos. Esta lista agrupa los recursos encontrados según su tipo, es decir, muestra una lista de recursos por cada tipo.
d. Una vez que se tenga la lista de recursos encontrados, se activa el proceso de selección a través de la utilización del enfoque cliente (*licitación*), sólo si el número de recursos o dueños de recursos es mayor que el de los clientes interesados. Este protocolo interactúa con los manejadores de recursos para seleccionar un recurso por tipo, buscando maximizar los beneficios del cliente. El proceso de selección se realiza utilizando el listado de requerimientos suministrado por el usuario, y los criterios orientados a satisfacer los parámetros de calidad de servicio:
    a. En Pl se genera un perfil (pliego de licitación) con los requerimientos del usuario. Este perfil es el patrón que siguen para buscar el mejor recurso entre la lista. El perfil consiste en la lista de recursos suministrados e indicadores de rendimiento: características del enlace de red, características del recurso en particular.
    b. El proceso de interacción, para los recursos con varios ofertantes se lleva a cabo de igual forma que en el enfoque cliente, esto es a través de la licitación. Al ser un proceso de licitación, los manejadores de servicios encontrados en la búsqueda inicial ofrecen su mejor propuesta para intentar ganar la buena pro del cliente, a través del cumplimiento de los requisitos, y de la minimización de costos.
e. Para los casos donde sólo existen un vendedor, o hay varios clientes concurrentemente solicitando un recurso en particular, el proceso de interacción para la negociación se cambia por el de las *subastas*, y sólo en los casos donde

exista un solo vendedor y un solo comprador la negociación se lleva a cabo con el precio inicial de venta.

f. La interacción (licitación, subasta o compra directa) finaliza cuando se ha encontrado un recurso que cumple con los criterios establecidos propios de cada caso (ver caso de los dos esquemas anteriores).
g. Una vez finalizada la selección de todos los recursos necesarios para cumplir con los requerimientos del cliente, o del dueño del recurso según sea el caso, se establece un acuerdo que consiste en indicarle a cada extremo de la negociación (cliente y recursos) la asignación respectiva. Una vez establecido dicho acuerdo con todos los involucrados, se realiza la contratación con todos ellos.
h. Después de realizar dicha contratación, se da la autorización al cliente para que utilice los recursos según lo acordado: precios, tiempo de uso, restricciones. Esto implica decirle al recurso la tarea que se le asignó. Dicho recurso, según sus mecanismos de planificación, gestionará el uso del recurso por parte del usuario. Esto lo desarrolla a través de su planificador a corto plazo, que actúa a partir de ese momento.

En la figura 4 puede resumirse de forma gráfica este planteamiento, donde se inicia un proceso en el enfoque cliente, y cuando surgen los casos donde aparecen un vendedor para un recurso, o varios clientes para ese recurso, el enfoque de negociación se cambia al enfoque recurso, dándole el matiz adaptativo que anteriormente se había comentado.

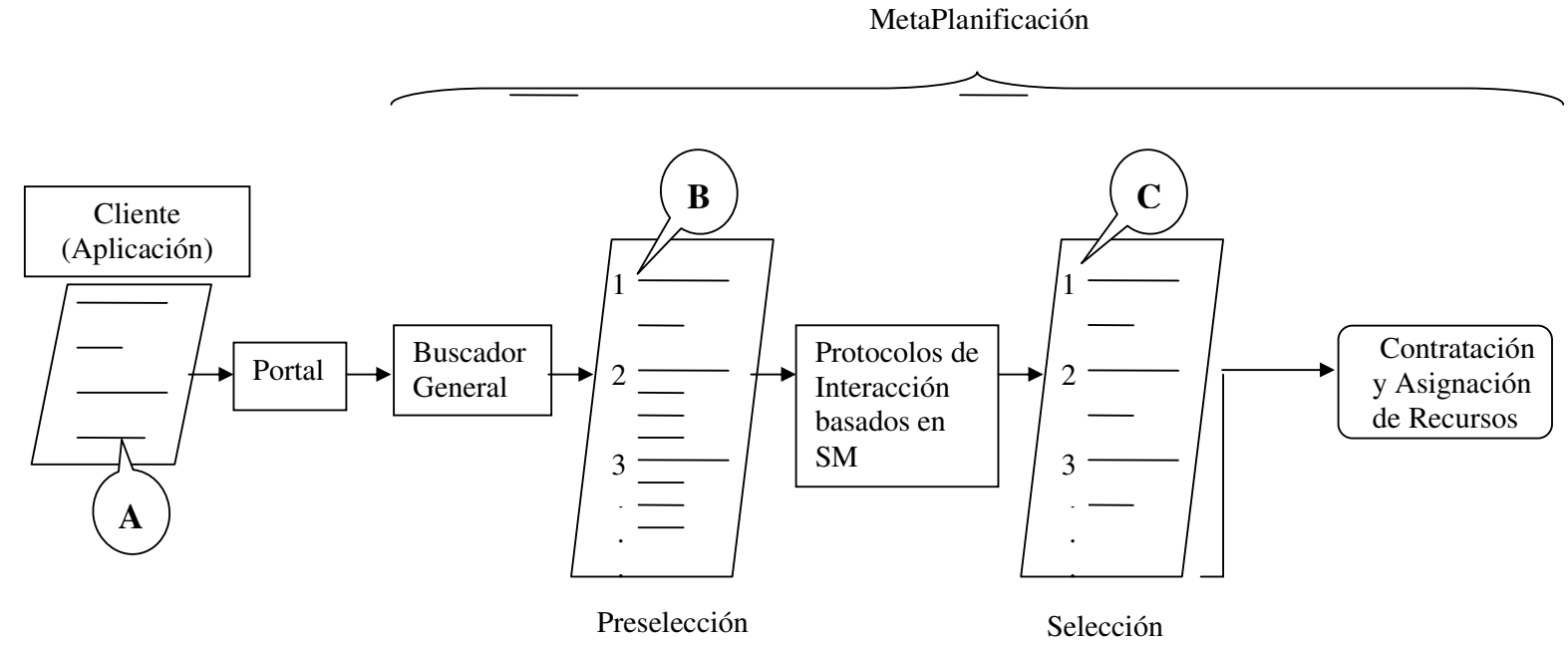
La figura 4 tiene gran similitud con la figura 1, la diferencia está que en el listado generado por el buscador general, sólo aparece un solo recurso para el reglón 1 de la parte B, lo cual quiere decir que sólo se encontró un vendedor para ese tipo de recurso, el cual, además, puede estar siendo requerido por otros clientes al mismo tiempo, y para este caso en particular, el protocolo de interacción cambia (se

adapta) hacia un tipo de negociación directa, o bajo el esquema de subastas, según sea el caso.

3.1.2. Arquitectura de Agentes

En las secciones anteriores se mostró el marco general del metaplanificador propuesto, en esta sección se muestra el protocolo en base a la representación de cada uno de los agentes que lo componen, y la forma en que ellos interactúan (se comunican). Para este propósito se utilizan las técnicas propuestas en [26] para la representación de protocolos de interacción basados en agentes. Este trabajo plantea una representación denominada AUML, que es una extensión de UML (Unified Modeling Language) a la cual incorpora la representación de agentes. Los esquemas de representación planteados están organizados en tres grupos, el primero está conformado por la representación de los protocolos vistos como un todo, de forma integral con todos sus componentes, para tal fin se utilizan las "plantillas" y los "paquetes". El segundo grupo procura capturar las interacciones o dinámicas entre agentes y utiliza dos esquemas pertenecientes a la familia de los diagramas de interacción: los diagramas de secuencia extendidos, también llamados diagramas de protocolo, y los diagramas de colaboración, ambos, por separado, pueden ser utilizados para representar a cualquier protocolo. Además, en el tercer conjunto, se utilizan estrategias de representación con mayores niveles de detalles, y pretenden capturar las dinámicas dentro de cada agente y también entre los agentes, tales como los diagramas de actividades y los diagramas de estados. Lo anterior se resume en que AUML ofrece una representación en tres capas de los protocolos de interacción basados en agentes. En cada capa se plantean distintos niveles de especificidad.

Dado que los diagramas de protocolo forman parte constitutiva de las plantillas, en este trabajo se utilizan ambos esquemas en una misma representación del protocolo de interacción propuesto. Además, para mostrar con mayor claridad lo planteado, se utiliza una representación con diagramas de colaboración, que es complementada con los diagramas de actividades que incrementan el nivel de detalle especificado.



A= Requerimientos del Cliente.
B= Lista de recursos encontrados agrupados por el tipo requerido.
C= Lista de recursos seleccionados por tipo requerido.

Figura 4: Enfoque híbrido de la Metaplanificación

Los diagramas de protocolos representan los procesos de comunicación (intercambio de mensajes) entre los agentes, e incluye la semántica de cada uno de los mensajes asociados, utilizando el formato incorporado por FIPA [12]. En estos diagramas se manejan dos ejes de referencia, en el eje horizontal se representa cada uno de los entes (agentes) involucrados en un proceso de comunicación, y en el eje vertical se representa el tiempo, que transcurre desde la parte superior del diagrama hacia la parte inferior. Cuando la representación se hace sin incluir instanciaciones del protocolo, es decir se presenta de forma abstracta o general, se está conformando la estructura de una plantilla.

En los diagramas de colaboración se representan a los agentes en cuadros o cajas unidas por líneas que constituyen las interacciones entre ellos. El orden secuencial o cronológico se especifica por la numeración que acompaña a cada mensaje, el cual se indica junto con cada una de las flechas que representan la dirección de las interacciones. Cuando es necesario especificar las actividades internas de un agente, se utilizan los diagramas de actividades, superpuestos al diagrama de colaboración.

A continuación se definen los tres enfoques (cliente, recurso e híbrido) a través de sus plantillas, es decir, a través de sus diagramas de protocolos (DP) y sus semánticas, no instanciadas. La forma de definir la semántica se hizo utilizando como guía a [12], donde se especifican cada uno de los mensajes utilizados (semántica) según un patrón preestablecido en este documento. Para el enfoque cliente se utilizó una adaptación del protocolo FIPA [12] sobre la subasta inglesa, que es el tipo de subasta comúnmente aplicado en el contexto económico.

Inicialmente se explican brevemente el significado general de los distintos mensajes utilizados para este tipo de representación, y luego a través de la estructura de la semántica establecida, se especifica cada mensaje para cada uno de lo enfoques, representados en los tres diagramas:

accept-proposal: Es una aceptación de propósito general de una propuesta (propose) que fue previamente enviada. El agente que envía la aceptación informa al agente que hizo la propuesta, que está dispuesto a desarrollar la acción planteada. Como parte de la aceptación se indican las condiciones bajo las cuales el agente está aceptando la propuesta.

cfp(call for proposal): Es una acción de propósito general que inicia un proceso de negociación haciendo una llamada o solicitud a uno o varios agentes para que éstos generen propuestas, con respecto a una acción planteada. Los parámetros del proceso de negociación se conocen con anticipación y explícitamente se colocan como parte del mensaje.

inform: Es una acción (mensaje) por medio de la cual el agente que la emite desea informar al receptor sobre el estatus de una propuesta que como emisor hizo con anterioridad.

not-understood: El emisor de este mensaje informa que no ha entendido el comunicado o mensaje anterior.

propose: Es una acción de propósito general para hacer una propuesta durante un proceso de negociación. Se propone desarrollar una acción dada enmarcada dentro de ciertas condiciones. Se deben conocer con anterioridad los parámetros del proceso de negociación, o el mensaje de la proposición los debe contener. El emisor de la propuesta informa al receptor que tiene la intención de desarrollar la acción una vez que las condiciones sean cumplidas, y el recibidor notifique al emisor su intención de desarrollar la acción.

reject-proposal: Es un acción de propósito general que rechaza una propuesta anteriormente enviada. El agente emisor del rechazo informa al receptor que no tiene la intención de desarrollar la acción bajo las condiciones dadas.

request: El agente emisor solicita al receptor el desarrollo de cierta acción. El contenido del mensaje es una descripción de la acción a ser desarrollada, en un lenguaje que el receptor entienda. La acción puede ser cualquiera que el receptor pueda desarrollar.

3.1.2.1. Enfoque Recurso

Este enfoque se representa a través de un diagrama de protocolo mostrado en la figura 5. El proceso de subasta puede ser visto cuando se le hace un seguimiento sistemático a los mensajes, considerando su contenido y su motivo de envío, lo cual se explica en su semántica explicada a continuación.

```
(request-1
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"comprar-servicios (recurso Y, oferta del corredor X)"
:language C)
```

Descripción: Este mensaje es del corredor hacia el coordinador, con el fin de notificarle su requerimiento por un recurso o servicio. Se especifica el lenguaje de implementación, que para el caso es el utilizado en la simulación con SimGrid. Esto se replica para el resto de los mensajes.

```
(request-2
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name buscador))
:content
"buscar-servicios (recurso Y)"
:language C)
```

Descripción: Este mensaje es la solicitud de búsqueda del coordinador al buscador, dada la solicitud por el recurso o servicio hecha previamente por el corredor.

```
(request-3
:sender (agent-identifier :name buscador)
:receiver (set (agent-identifier :name manejador))
:content
"solicitud-servicios (recurso Y)"
:language C)
```

Descripción: Este mensaje es la solicitud que hace el buscador a los manejadores de recursos para saber si están en la disposición de ofrecer el recurso o servicio solicitado.

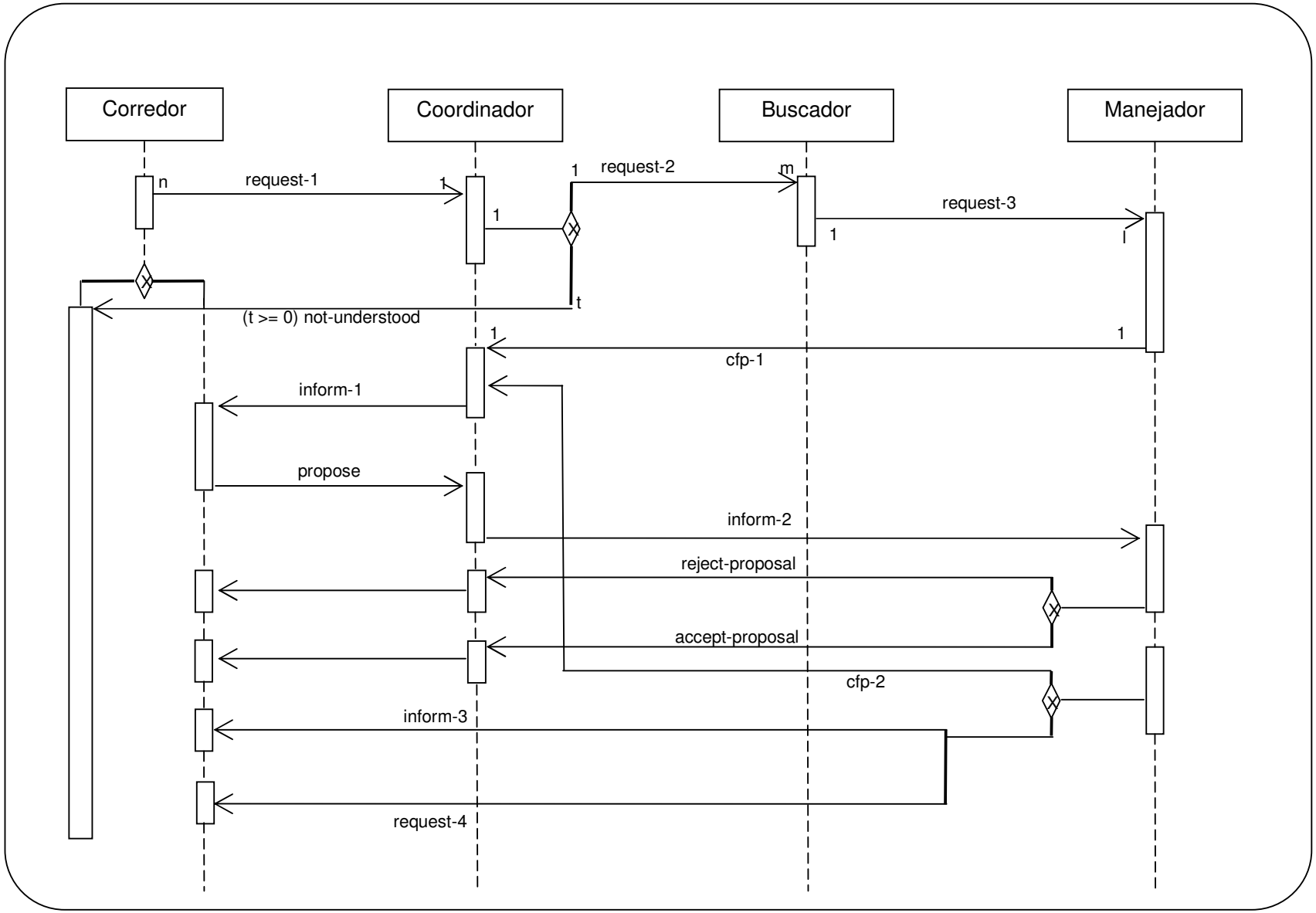


Figura 5: Diagrama de Protocolo - Enfoque Recurso

```
(request-4
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name corredor))
:content
"solicitud-información-adicional (recurso Y, oferta del
corredor X, parámetro requerido para Y)"
:language C)
```

Descripción: Este mensaje se envía al final de la negociación, previa aprobación de la contratación, para solicitar, si es necesario, datos adicionales del cliente.

```
(cfp-1
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordiandor)
(vender recurso Y))
(para Y (precio Y > C)"
:language C)
```

Descripción: Este mensaje contiene una invitación o llamado a hacer una propuesta de parte del manejador, hacia el coordinador, para éste le informe a los corredores que pueden hacer sus primeras ofertas o propuestas.

```
(cfp-2
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name interaccion))
:content
"((action (agent-identifier :name coordinador)
(vender recurso Y))
(para Y (precio Y > C+M)"
:language C)
```

Descripción: Este mensaje se utiliza, en términos de los propósitos de la subasta, para solicitar nuevas propuestas a los corredores a través del coordinador. Con esto se genera el ciclo de ofertas de la subasta al permitir nuevas propuestas de los corredores.

```
(propose
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"(action (interacción (comprar recurso Y, oferta a Y)))
:in-reply-to cfp-1 o cfp-2
:language C)
```

Descripción: Este mensaje contiene la propuesta de negociación hecha por parte del corredor hacia el manejador por la vía del coordinador.

```
(reject-proposal
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(venta recuros Y))
(oferta del corredor X)
(oferta-muy-baja Y))"
:in-reply-to cfp-1 o cfp-2
:language C)
```

Descripción: Este mensaje es para solicitarle al coordinador que le informe al corredor que su propuesta ha sido rechazada.

```
(accept-proposal
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name interaccion))
:in-reply-to propose
:content
"((action (agent-identifier :name interaccion)
(stream-content acepta-oferta))
(B (agent-identifier :name interaccion)
(ready corredorX)))"
:language C)
```

Descripción: Este mensaje es para solicitarle al coordinador que le informe al corredor que su propuesta ha sido aceptada.

```
(inform-1
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name corredor))
:content
"cfp-1 (Y, precio Y)"
:language C)
```

Descripción: Este mensaje es para que el coordinador le informe al corredor sobre el llamado a hacer una propuesta hecha por el manejador.

```
(inform-2
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name manejador))
:content
"propose (corredor X, oferta de corredor X)"
:language C)
```

Descripción: Este mensaje lo utiliza el coordinador para informarle al manejador sobre la propuesta hecha por el corredor.

```
(inform-3
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name corredor))
:content
"venta-hecha (Y, precio Y)"
:language C)
```

Descripción: Este mensaje lo utiliza el manejador para informarle al corredor el estatus de la aceptación de su propuesta.

```
(not-understood
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name corredor))
:content
```

"no entiendo (corredor X, request-1)"
:language C)

Descripción: Este tipo de mensaje lo utiliza el manejador para indicarle al corredor que no pudo procesar su solicitud.

3.1.2.2. Enfoque Cliente

La figura 6 representa el diagrama de protocolo establecido para este enfoque, en la cual se esquematiza sistemáticamente el proceso de licitación, que puede ser visto siguiendo la estructura de mensajes, conociendo su contenido y su justificación de envío explicada en la semántica a continuación mostrada:

(request-1
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"comprar-servicios (recurso Y, oferta del corredor X)"
:language C)

Descripción: Este mensaje lo utiliza un corredor para indicarle al coordinador sobre su requerimiento por un recurso o servicio.

(request-2
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name buscador))
:content
"buscar-servicios (recurso Y)"
:language C)

Descripción: Este mensaje lo utiliza el coordinador para solicitarle al buscador que inicie el proceso de búsqueda según el requerimiento enviado por el corredor.

(request-3
:sender (agent-identifier :name buscador)
:receiver (set (agent-identifier :name manejador))
:content
"solicitud-servicios (recurso Y)"

```
:language C)
```

Descripción: Este mensaje lo utiliza el buscador para solicitarle al manejador que le indique su disponibilidad por el recurso o servicio requerido.

```
(request-4
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name manejador))
:content
"solicitud-información-adicional (recurso Y, oferta del
corredor X, parámetro adicional requerido para compra
de Y)"
:language C)
```

Descripción: Este mensaje lo utiliza el corredor para solicitarle, si es necesario, información adicional al manejador, dada la aceptación de la contratación.

```
(cfp
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(comprar recurso (Y))
(para Y (precio Y < C)"
:language C)
```

Descripción: Este mensaje se lo envía el corredor al coordinador con la intención de hacer un llamado al manejador para que genere una propuesta de venta del recurso o servicio. En él se especifican los parámetros que debe contener tal propuesta, es decir, el pliego de licitación.

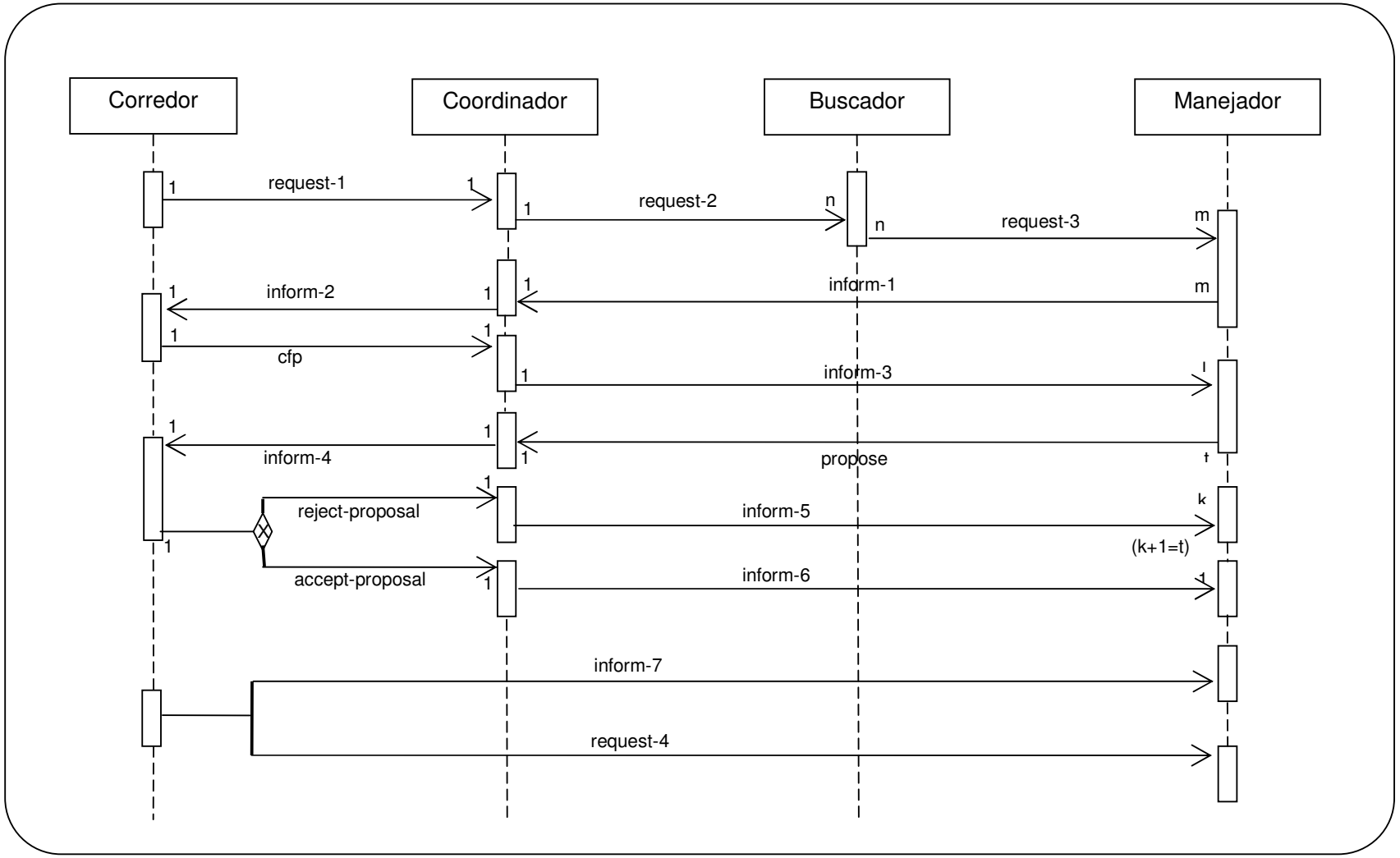


Figura 6: Diagrama de Protocolo - Enfoque Cliente

```
(propose
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"(action (interacción (comprar recurso Y, oferta a Y)))
:in-reply-to cfp
:language C)
```

Descripción: Este mensaje lo utiliza el manejador para hacer su propuesta.

```
(reject-proposal
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(compra recurso Y))
(cfp del corredor X)
(precio-muy-alto de Y))"
:in-reply-to propose
:language C)
```

Descripción: Este mensaje lo utiliza el corredor para rechazar una propuesta del manejador, lo cual lo tramita a través del coordinador.

```
(accept-proposal
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name interaccion))
:in-reply-to propose
:content
"((action (agent-identifier :name interaccion)
(acepta-propuesta(corridor X))"
:language C)
```

Descripción: Este mensaje lo utiliza el corredor para informar la aceptación de una propuesta previamente enviada.

```
(inform-1
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"venta-recurso (Y)"
:language C)
```

Descripción: Con este mensaje el manejador le indica al corredor su disponibilidad para vender el recurso o prestar el servicio.

```
(inform-2
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name corredor))
:content
"venta-recurso (recurso Y, manejador del recurso Y)"
:language C)
```

Descripción: Con este mensaje el coordinador le informa al corredor la disposición del manejador.

```
(inform-3
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name manejador))
:content
"cfp (corredor X, oferta del corredor X al recurso Y)"
:language C)
```

Descripción: Con este mensaje el coordinador le informa al manejador sobre el llamado a licitación hecho por parte del corredor.

```
(inform-4
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name corredor))
:content
"propose(recurso Y, precio del recurso Y, manejador del
recurso Y)"
:language C)
```

Descripción: Con este mensaje el coordinador le informa al corredor sobre la propuesta hecha por el manejador.

```
(inform-5
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name manejador))
:content
"reject-proposal (corredor X, recurso Y)"
:language C)
```

Descripción: Con este mensaje el coordinador le informa al manejador que el corredor ha rechazado su propuesta.

```
(inform-6
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name manejador))
:content
"accept-proposal (corredor X, oferta del corredor X al
recurso Y)"
:language C)
```

Descripción: Con este mensaje el coordinador le informa al manejador que el corredor ha aceptado su propuesta.

```
(inform-7
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name manejador))
:content
"coordinacion (corredor X, inicio de ejecución)"
:language C)
```

Descripción: Con este mensaje el corredor le indica al manejador el estatus de la contratación.

3.1.2.3. Enfoque Híbrido

La Figura 7 es el diagrama de protocolo para este enfoque. Al igual que en los dos diagramas anteriores el proceso se puede entender haciendo un seguimiento

sistemático de los mensajes, considerando su contenido y justificación de envío, explicados en la semántica mostrada a continuación.

```
(request-1
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"comprar-servicios (recurso Y, oferta del corredor X)"
:language C)
```

Descripción: Con este mensaje el corredor le envía al coordinador su requerimiento.

```
(request-2
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name buscador))
:content
"buscar-servicios (recurso Y)"
:language C)
```

Descripción: Por medio de este mensaje el corredor le solicita al buscador que consiga los manejadores que pueden satisfacer el requerimiento del corredor.

```
(request-3
:sender (agent-identifier :name buscador)
:receiver (set (agent-identifier :name manejador))
:content
"solicitud-servicios (recurso Y)"
:language C)
```

Descripción: Con este mensaje el buscador le solicita al manejador información sobre su disponibilidad para ofrecer el recurso o servicio buscado.

```
(request-4
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name manejador))
:content
```

"solicitud-información-adicional (recurso Y, oferta del corredor X, parámetro adicional requerido para compra de Y)"
:language C)

Descripción: Para el caso de licitación, con este mensaje se solicita información adicional al manejador, si el caso lo amerita.

(request-5
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name corredor))
:content
"solicitud-información-adicional (recurso Y, oferta del corredor X, parámetro adicional requerido para vender a Y)"
:language C)

Descripción: Para el caso de subasta, con este mensaje se lo solicita al corredor información adicional si el caso lo amerita.

(cfp-1
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(vender recurso Y))
(para Y (precio Y > C)"
:language C)

Descripción: Para los casos donde exista un mayor número de clientes con respecto al número de manejadores, se establece el caso subasta (enfoque recurso), donde el manejador le envía un llamado a hacer propuestas (ofertas) a los corredores por medio de los coordinadores.

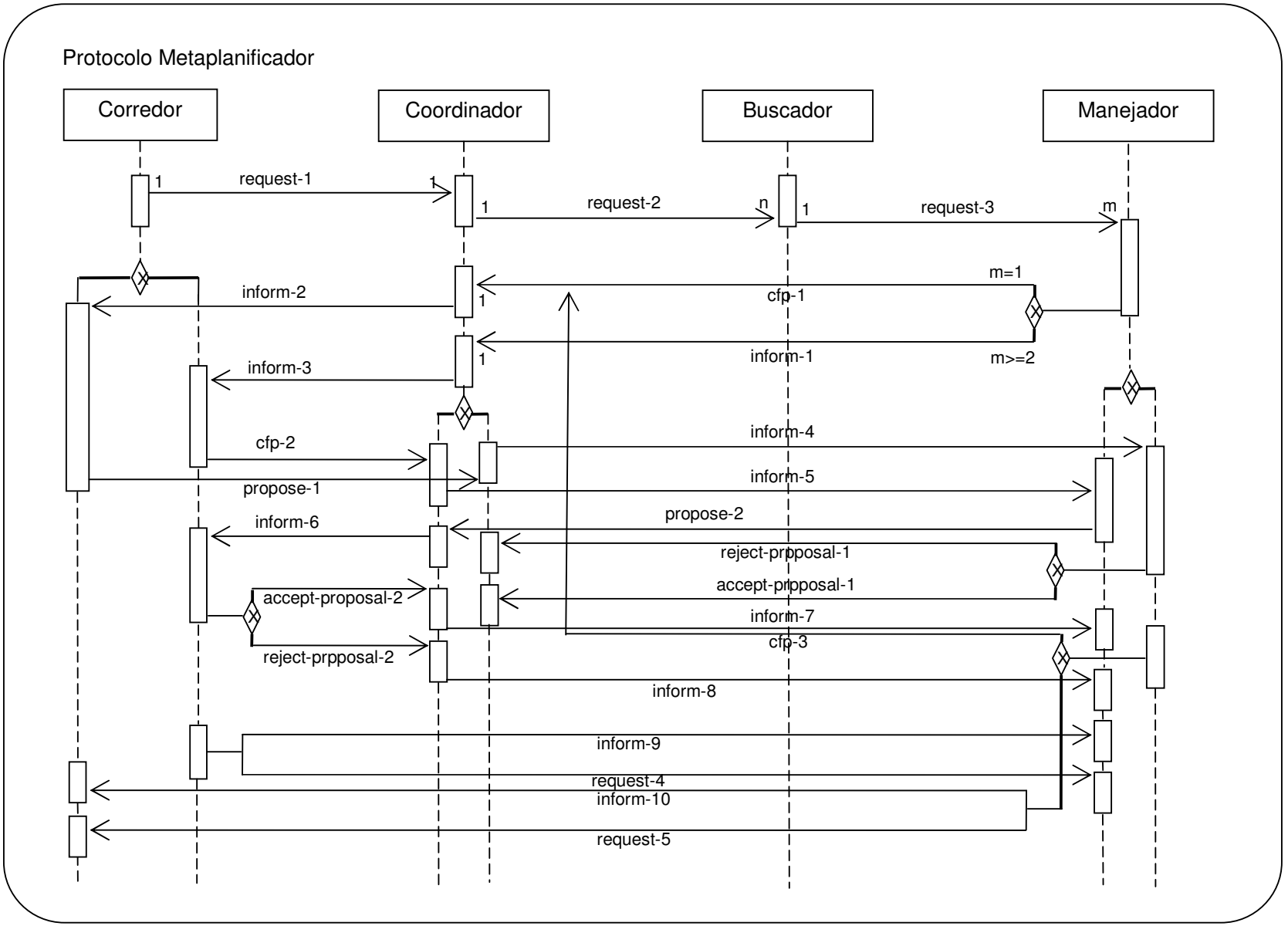


Figura 7: Diagrama de Protocolo - Enfoque Híbrido

```
(cfp-3
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(vender recurso Y))
(para Y (precio Y > C+M)"
:language C)
```

Descripción: Al igual que el caso anterior, este mensaje es parte de un proceso de subasta (enfoque recurso), el cual permite desarrollar el ciclo de la subasta al permitir realizar otros llamados de propuestas adicionales, con lo cual los cliente pueden realizar propuestas para aumentar sus ofertas en varias oportunidades.

```
(cfp-2
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(comprar recurso (Y))
(para Y (precio Y < C)"
:language C)
```

Descripción: Este mensaje se activa si existe mayor cantidad de manejadores que de corredores, con relación a un recurso o servicio en particular. Lo que implica que se establece un proceso de licitación, y con el cfp-2 el cliente le hace el llamado al manejador para que haga su propuesta.

```
(propose-1
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"(action (interacción (comprar recurso Y, oferta a Y)))
:in-reply-to cfp-1 o cfp-3
:language C)
```

Descripción: Este mensaje lo envía un corredor al coordinador, previamente establecida una subasta, para hacer su propuesta (oferta) al manejador.

```
(propose-2
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"(action (interacción (comprar recurso Y, oferta a Y)))
:in-reply-to cfp-2
:language C)
```

Descripción: Este mensaje se lo envía un manejador al corredor, previamente establecido el proceso de licitación, para hacer su propuesta ante el pliego de licitación.

```
(reject-proposal-1
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(venta recuros Y))
(oferta del corredor X)
(oferta-muy-baja Y))"
:in-reply-to propose-1
:language C)
```

Descripción: Este mensaje lo envía un manejador al coordinador, previamente establecida una subasta, para rechazar una propuesta (oferta) del corredor.

```
(accept-proposal-1
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:in-reply-to propose-1
:content
"((action (agent-identifier :name coordinador)
(stream-content acepta-oferta))
```

```
(B (agent-identifier :name coordinador)
(ready corredorX)))"
:language C)
```

Descripción: Este mensaje lo envía un manejador al coordinador, previamente establecida una subasta, para aceptar una propuesta (oferta) del corredor.

```
(reject-proposal-2
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:content
"((action (agent-identifier :name coordinador)
(compra recurso Y))
(cfp del corredor X)
(precio-muy-alto de Y))"
:in-reply-to propose-2
:language C)
```

Descripción: Este mensaje lo envía un corredor al coordinador, previamente establecida una licitación, para rechazar una propuesta del manejador.

```
(accept-proposal-2
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name coordinador))
:in-reply-to propose-2
:content
"((action (agent-identifier :name coordinador)
(acepta-propuesta(corridor X))"
:language C)
```

Descripción: Este mensaje lo envía un corredor al coordinador, previamente establecida una licitación, para aceptar una propuesta del manejador.

```
(inform-1
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name coordinador))
:content
```

"venta-recurso (Y)"

:language C)

Descripción: Para el caso de una licitación, el manejador le envía al coordinador su disponibilidad para satisfacer el requerimiento del corredor.

(inform-2

:sender (agent-identifier :name coordinador)

:receiver (set (agent-identifier :name corredor))

:content

"cfp-1 (Y, precio Y)"

:language C)

Descripción: Para el caso de una subasta, este mensaje se lo envía el coordinador al corredor para informarle sobre el llamado de propuesta que hizo el manejador.

(inform-3

:sender (agent-identifier :name coordinador)

:receiver (set (agent-identifier :name corredor))

:content

"venta-recurso (recurso Y, manejador del recurso Y)"

:language C)

Descripción: Para el caso de una licitación, el coordinador le envía al corredor la información sobre la disponibilidad del manejador de satisfacer lo requerido.

(inform-4

:sender (agent-identifier :name coordinador)

:receiver (set (agent-identifier :name manejador))

:content

"propose (corredor X, oferta de corredor X)"

:language C)

Descripción: Para el caso de una subasta, el coordinador le envía el manejador la información sobre la propuesta (oferta) que hizo el corredor.

(inform-5

:sender (agent-identifier :name coordinador)

```
:receiver (set (agent-identifier :name manejador))
:content
"cfp (corredor X, oferta del corredor X al recurso Y)"
:language C)
```

Descripción: Para el caso de una licitación, el coordinador le envía al manejador la información sobre el llamado a propuesta que hizo el corredor.

```
(inform-6
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name corredor))
:content
"propose(recurso Y, precio del recurso Y, manejador del
recurso Y)"
:language C)
```

Descripción: Para el caso de licitación, este mensaje se lo envía el coordinador al corredor para informarle sobre la propuesta hecha por el manejador.

```
(inform-7
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name manejador))
:content
"reject-proposal (corredor X, recurso Y)"
:language C)
```

Descripción: Para el caso de licitación, este mensaje se lo envía el coordinador al manejador para informarle que el corredor a rechazado su propuesta.

```
(inform-8
:sender (agent-identifier :name coordinador)
:receiver (set (agent-identifier :name manejador))
:content
"accept-proposal (corredor X, oferta del corredor X al
recurso Y)"
:language C)
```

Descripción: Para el caso de licitación, este mensaje se lo envía el coordinador al manejador para informarle que el corredor a aceptado su propuesta.

```
(inform-9
:sender (agent-identifier :name corredor)
:receiver (set (agent-identifier :name manejador))
:content
"coordinacion (corredor X, inicio de ejecución)"
:language C)
```

Descripción: Para el caso de licitación, este mensaje se lo envía el corredor al manejador, si necesita información adicional para la contratación ya establecida.

```
(inform-10
:sender (agent-identifier :name manejador)
:receiver (set (agent-identifier :name corredor))
:content
"venta-hecha (Y, precio Y)"
:language C)
```

Descripción: Para el caso de subasta, este mensaje se lo envía el manejador al corredor, si necesita información adicional para la contratación ya establecida.

Para complementar la representación de la propuesta hecha, y en vista de que el enfoque híbrido fue el que se implementó en la simulación, dado que engloba los dos enfoques anteriores, a continuación se presenta un diagrama de colaboración acompañado por un diagrama de actividades, con los cuales se esquematiza el protocolo en su versión híbrida, detallando las actividades que desarrolla el coordinador en su comunicación con los demás agentes, y representando las interacciones como tal. La figura 8 muestra estos diagramas, donde en el diagrama de colaboración las flechas indican el sentido de las interacciones, que se acompañan de un listado de los mensajes asociados a cada interacción, presentados con una numeración que indica el orden cronológico o secuencial de los envíos.

Los casos de subasta se indican con una "s", por ejemplo, en la figura 8, el mensaje ***3s:cfp-1***, indica que para el caso subasta, en el paso ***3*** se envió el mensaje ***cfp-1*** del

manejador al coordinador. Los casos de licitación se indican con una “l”. Además, es importante destacar que se presentan dos roles para los corredores, y dos roles para los manejadores, el cambio de rol se ejecuta dependiendo del caso que se establezca, ya sea el de subasta o el de licitación, es decir, cada rol indica sus interacciones según cada caso. El diagrama de actividades se establece debajo del agente coordinador, señalado por las líneas horizontales consecutivas. En este diagrama se muestran las actividades del coordinador desde que recibe la solicitud de los corredores, hasta establecer un ganador, ya sea para una subasta o para una licitación. La explicación o semántica de los mensajes es la misma que la utilizada para el diagrama de protocolo del enfoque híbrido. Para evitar sobrecargar el diagrama, se obvian las interacciones asociadas a los buscadores, ya que no son parte real del protocolo propuesto.

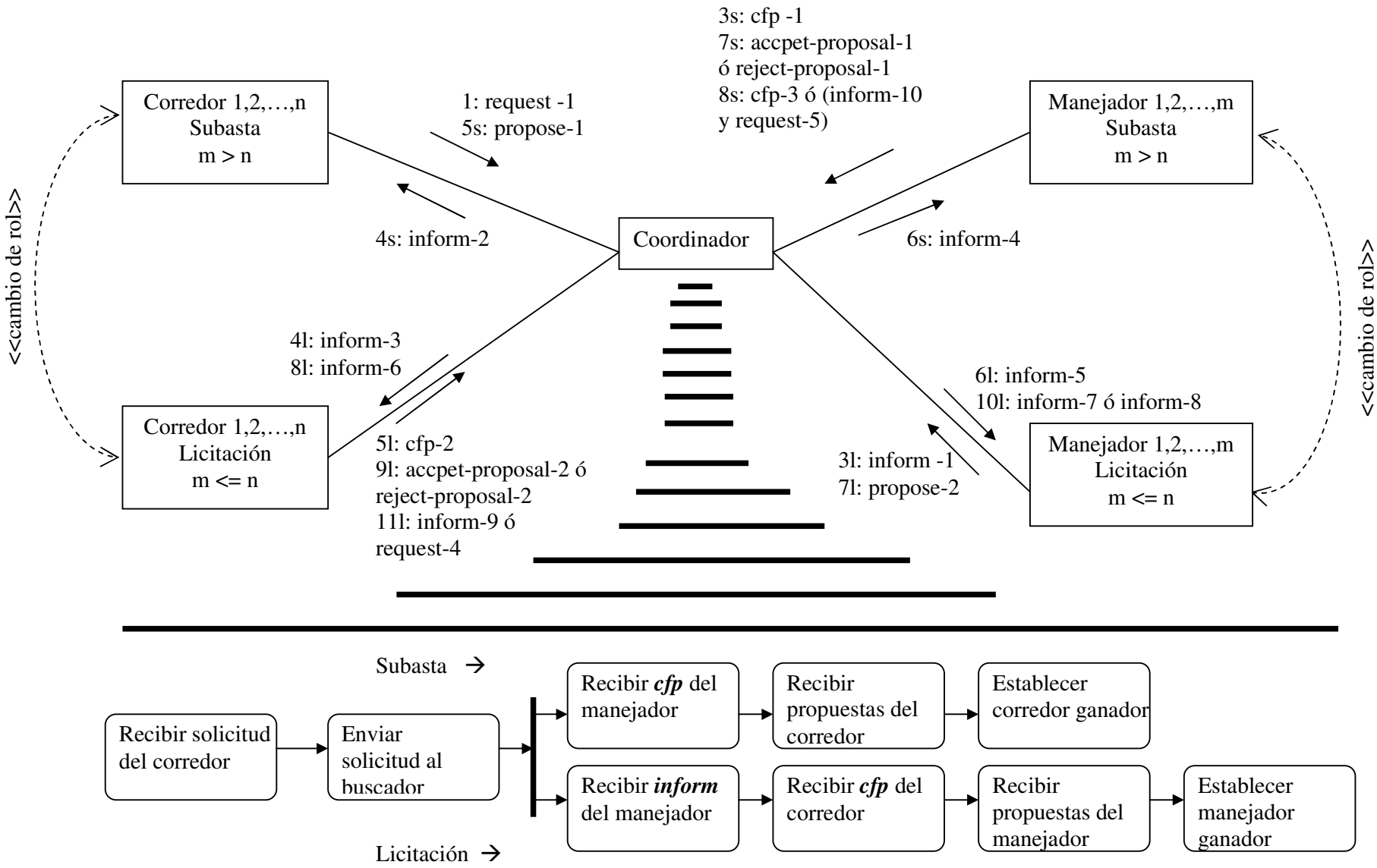


Figura 8. Diagrama de colaboración y de actividades del Enfoque Híbrido

CAPÍTULO IV
PRUEBAS DE LA PROPUESTA

4.1. MODELADO Y SIMULACIÓN DEL PROTOCOLO EN SIMGRID

Para probar el protocolo propuesto, se utilizó como estrategia de pruebas el modelado y la simulación. Como se mencionó anteriormente, entre las ventajas que se obtienen al simular está la de poder tener ambientes controlados, lo cual permite hacer diferentes pruebas con resultados válidos en tiempos mucho más cortos de lo que se pueden establecer en las implementaciones reales.

Una de las características que se debe destacar en este tipo de estrategia, es que el modelo es una representación simplificada, con cierto nivel de abstracción, en comparación a la realidad. Por este motivo, nuestro modelo planteado representa al protocolo propuesto simplificando gran parte de su estructura. Como primera observación en relación a esto, se debe comentar que el modelo no incluye a los buscadores presentes en los entornos reales, dado que no son parte constitutiva de la propuesta. Tampoco se consideran los usuarios y dueños de recursos como entes físicos, y sólo se incluyen en la simulación las actividades e intercambios desarrollados por sus representantes digitales: los agentes.

De esta forma, el modelo está constituido por los agentes involucrados en la propuesta: corredor, coordinador, y manejador. Cada uno se modela a través de sus características principales que son necesarias para representar la dinámica completa de protocolo.

La herramienta de simulación utilizada fue el SimGrid, que es una librería hecha en el lenguaje C, ver [7, 8], la cual permite simular algoritmos de planificación distribuidos. Particularmente se utilizó el módulo denominado MetaSimGrid o MSG, diseñado para realizar simulaciones de índole general. Este módulo posee características particulares que son necesarias entenderlas y aplicarlas para poder generar las simulaciones requeridas. Inicialmente, el MSG utiliza varios componentes básicos para modelar algoritmos de planificación. Tales componentes se definen a continuación:

- Proceso: Está definido como un código en ejecución, que incluye data privada, ejecutado sobre alguna localidad (host). En los términos de este trabajo, es el código en ejecución de un agente. Representa el corazón del

simulador, dado que las decisiones sobre la planificación se hacen sobre los procesos.

- Localidad o Host: Es cualquier lugar donde puede correr un proceso. Puede representar un recurso físico con ciertas capacidades de cómputo, algunas estructuras de almacenamiento temporal o *mailboxes* (permiten el manejo de procesos y su comunicación con otras localidades), y alguna data privada. Este componente representa a los agentes bajo el enfoque de sistemas multiagentes, ya que cada host tiene un código asociado, y se pueden configurar los niveles requeridos de autonomía.
- Tarea: De forma abstracta se define por una cantidad de cómputo necesaria para su ejecución, un tamaño en bytes relacionado al ancho de banda que consume su transmisión, junto con alguna data privada. Las tareas son las que se utilizan en este trabajo para representar las actividades o solicitudes generadas en la GRID, ya sean de los corredores, del coordinador, o de los manejadores. Cada proceso relacionado a un agente puede generar varias tareas. Por ejemplo, si un cliente genera una solicitud en la que se requiera algún tipo de recurso, ese cliente crea una tarea con los datos de la solicitud, y después la envía al coordinador. En el envío el número de bytes asignados para esa tarea consumirán el ancho de banda de la conexión asignada, y cuando el coordinador la recibe la debe procesar, utilizando la cantidad de cómputo que previamente se declaró que debía consumir esa tarea. Este cómputo es el requerido en el proceso de negociación (planificación), y no tiene relación con los procesos de cómputo sobre los recursos finales, los cuales no se están modelando.
- Enlace: Dentro de un contexto de una organización lógica de redes, y no real, representa un conjunto de líneas de comunicación que caracterizan a un conjunto de enlaces físicos de red. Se utilizan para modelar de forma indirecta los procesos de enrutamiento.
- Canal: Tiene el mismo significado que un puerto dentro del contexto de redes bajo la arquitectura TCP/IP (son como puertos abiertos en cada localidad o host, y no representan "sockets").

El MSG requiere que se defina el código asociado a cada agente (host) a través de funciones representadas en el lenguaje C, cada una de estas funciones puede recibir un conjunto de parámetros de entrada, necesarios para completar las características que se definen para generar las instancias de cada agente, lo que permite la ejecución de su código, que es lo que se interpreta como un proceso. En cada proceso se genera la creación, envío, recepción, ejecución y finalización de tareas, que como se comentó anteriormente, representan las actividades de cómputo y de envío de mensajes, para la interacción.

Específicamente para este trabajo se creó un archivo escrito en lenguaje C (ver anexo A), en el cual se establecen cada una de las funciones que definen a cada agente: corredor, coordinador y manejador. Esta arquitectura de agentes se representa en la figura 9, donde se plantea el uso de varios agentes corredores, cada un utilizando una instancia de la función para agentes corredores, un solo coordinador, que utiliza una instancia de la función para agente coordinador, y varios agentes manejadores, que utilizan instancias de la función definida para los agentes manejadores.

Para la *función relacionada al corredor*, se programaron las instrucciones necesarias para la creación de los mensajes iniciales, que incluyen los datos requeridos para establecer los procesos de subasta o licitación, los cuales son: montos iniciales, mínimos y máximos a ofrecer por el producto o servicio solicitado, y el tipo de producto o servicio solicitado. Para los efectos de simplificación y construcción del modelo, cada tipo de servicio o producto se representa por un número natural, iniciando desde 1, hasta la cantidad de tipos de recursos que puede solicitar un corredor. Cada corredor puede generar y enviar varias solicitudes para varios tipos de recursos. Para realizar cada solicitud debe generar y enviar una tarea que le permita hacer tal requerimiento al coordinador, cada una de estas tareas tendrán asociados los datos correspondientes al ancho de banda que consume esa tarea al ser enviada, y la capacidad de cómputo que requiere para ser procesada una vez que la reciba el coordinador, es decir, el cómputo requerida será procesado en el lado del coordinador. El conjunto de parámetros de entradas es: montos de la oferta inicial, máxima y mínima, número de solicitudes, cantidad en bytes del estimado que puede consumir cada tarea asociada al envío de un mensaje, cantidad en flops que puede consumir cada tarea generada asociada a cómputos, nombre del coordinador con el cual se negociará. Adicional a esto, en esta

función del agente corredor, se incluyen un conjunto de instrucciones destinadas a recibir los mensajes enviados por el coordinador.

En cuanto a la generación de solicitudes (tareas asociadas a las solicitudes generadas por los corredores), se estableció la utilización de métodos estocásticos, y específicamente la generación de tareas se representó a través de procesos Poison [18], calculando los tiempos entre cada tarea o solicitud por medio de números probabilísticos que siguen una Distribución Exponencial. Para representar esta distribución en el lenguaje C, se utilizó el método inverso [18], a través del cual se generan números con distribución exponencial, utilizando la función inversa de dicha distribución que utiliza como parámetros de entrada números con distribución uniforme generados en el rango {0,1}.

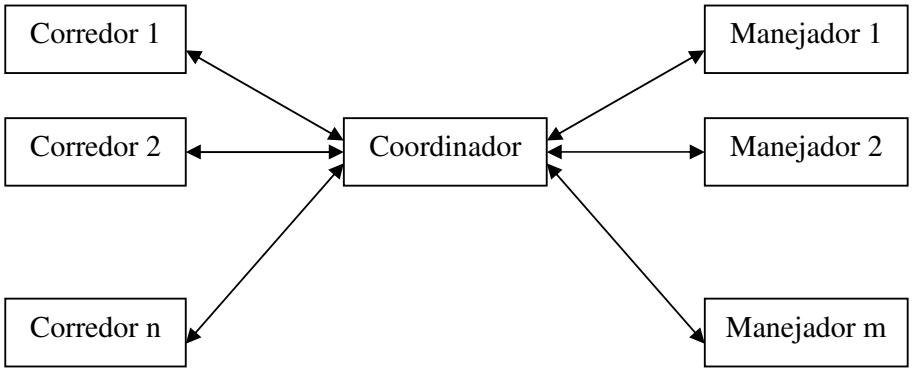


Figura 9. Arquitectura de Agentes en MSG

En la función del agente coordinador se programaron cuatro bloques específicos: el primero está orientado a la recepción de los mensajes enviados por los corredores, el segundo es para el envío de solicitudes o requerimientos a los manejadores y la recepción de las respuestas a estos envíos por parte de los manejadores, el tercer bloque es el corazón de la planificación, ya que decide si se establece un caso de subasta o de licitación, y dependiendo de cada caso, determina a los ganadores en cada tipo de proceso, según los datos enviados por los corredores y los manejadores. El cuarto bloque que la función del agente coordinador posee, está destinado a enviar los resultados de las asignaciones a los agentes corredores y manejadores.

Tal como se mencionó anteriormente, se modeló y simuló sólo el enfoque híbrido, ya que contiene los dos enfoques anteriores, y además es el que se implementaría en un ambiente real, por poseer características similares a los procesos de negociaciones presentes en la GRID. Si el número de manejadores es inferior al número de corredores, en relación a las solicitudes asociadas a un tipo de recurso, el planificador establecerá un proceso de subasta, en caso contrario, establecerá un proceso de licitación. La asignación se hace de la siguiente manera: si existen "m" manejadores, y "n" corredores, para el caso de subasta, los "m" manejadores (sus recursos asociados) serán asignados a los primeros "m" corredores que hayan ofrecido el monto mayor por ese tipo de producto o servicio. El resto de los "n-m" corredores se quedan si asignación para su solicitud, y se les informa al respecto. Para los casos de licitación, donde "n<m", a los "n" corredores se les asigna los "n" primeros manejadores (sus recursos asociados), que hayan ofrecido el menor costo del producto o servicio.

Para poder establecer el proceso de asignación, dentro del coordinador se utiliza una estructura de datos (una lista de listas), que contiene la información de todas las solicitudes hechas por los corredores, según su tipo, y los manejadores que respondieron informando que están dispuestos a surtir ese producto o prestar ese servicio. Para los efectos de una representación gráfica, esta estructura se muestra en la figura 10, en la cual se presenta la lista de listas, donde cada nodo de la lista general está asociado a un tipo distinto de recurso solicitado y generado por cada corredor, y ese nodo apunta o contiene dos listas adicionales, una de ellas contiene un conjunto de nodos que representan a cada uno de los corredores que solicitaron ese tipo de recurso o servicio, y en la otra lista, cada nodo representa a cada uno de los manejadores que están dispuestos a prestar ese tipo de servicio o a ofrecer ese tipo de producto.
Los parámetros de entrada de la función del agente coordinador son: número de corredores a atender, número de manejadores, listas de los nombres de los corredores, lista de los nombres de los manejadores, cantidad en bytes que consume el envío de cada tarea asociada a un mensaje, cantidad en flops que consume cada tarea asociada a cómputos.

Con respecto a *la función del agente manejador*, ésta está dividida en dos bloques, el primero se orienta a la recepción de las solicitudes enviadas por el coordinador, el segundo se utiliza para enviar la información relacionada al tipo de recurso que ese

manejador ofrece, tal como el costo inicial, el costo mínimo, y el tipo de recurso. Estos son los datos necesarios para establecer los procesos de subasta o licitación. Los costos pueden representar en términos reales a cualquier otra medida o información que se pueda establecer o requerir por parte de los manejadores, o hacia ellos mismos. Los parámetros de entrada de esta función son: número de recursos o servicio a ofrecer, nombre del corredor con el cual se negociará. Con lo que respecta a los tipos de recursos y sus costos iniciales, y costos mínimos, se generan de forma aleatoria (distribución uniforme de 0 al número de recursos introducido como parámetro) para darle una estructura dinámica al modelo planteado.

El esquema anterior representa cada iteración para cada período de tiempo particular, es decir, en la secuencia temporal, el proceso que se representa es el que corresponde a un período de tiempo. Las solicitudes que quedan pendientes en ese período de tiempo se supone que son consideras para los siguientes períodos de tiempo. Sólo se implementa la ejecución de períodos de tiempo individuales.

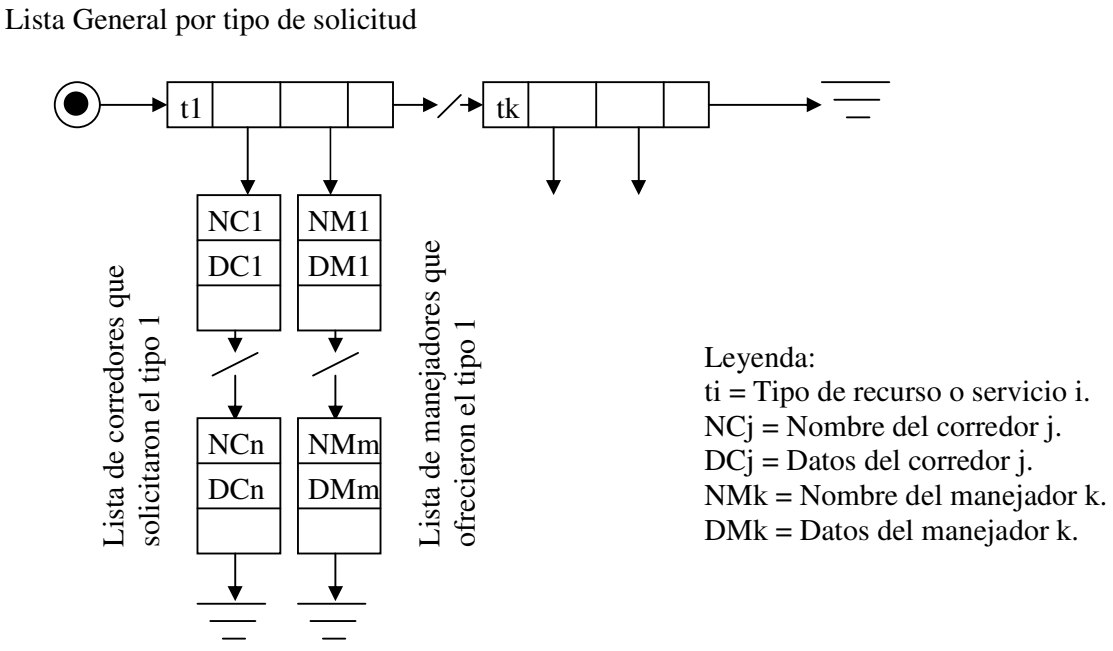


Figura 10. Estructura que almacena los tipos de recursos solicitados

Los parámetros de entrada de cada una de las instancias de los corredores, del coordinador, y de los manejadores se establecen en un archivo adicional, representado en formato XML, denominado *archivo de aplicación*. A continuación se muestra una plantilla genérica de dicho archivo.

```
<process host="nombre-corredor" function="corredor">
   <argument value="nombre-coordinador"/>
   <argument value="número-solicitudes"/>
   <argument value="requerimiento-de-computo-de-cada-tarea-
    en-flops"/>
   <argument value="tamaño-en-bytes-de-cada-tarea"/>
</process>

<process host="nombre-coordinador" function="coordinador">
     <argument value="número-de-corredores"/>
     <argument value="número-de-manejadores"/>
     <argument  value=  "requerimiento-de-computo-de-cada-
      tarea-en-flops"/>
     <argument value="tamaño-en-bytes-de-cada-tarea"/>
     <argument value="nombre-corredor-1"/>
     <argument value="nombre-corredor-n"/>
     <argument value="nombre-manejador-1"/>
     <argument value="nombre-manejador-m"/>
</process>

<process host="nombre-manejador" function="manejador">
     <argument value="número-de-recursos-ofrecidos"/>
     <argument     value="requerimiento-de-computo-de-cada-
      tarea-en-flops"/>
     <argument value="tamaño-en-bytes-de-cada-tarea"/>
</process>
```

Para establecer las características del modelo de red (arquitectura y topología) sobre la cual se implementa una GRID y su algoritmo de planificación, en otro archivo

adicional, en formato XML, denominado *archivo de plataforma*, se representa lo siguiente:

- Nombre y capacidad de cada uno de las computadoras donde se ejecutan los agentes corredores, coordinador, y manejadores. Sus capacidades son expresadas en número de instrucciones por segundo (flops).
- Nombre o número, ancho de banda (bytes por segundo), y latencia (segundos) de cada uno de los enlaces de red.
- Nombres del computador de origen y de destino establecidos para cada proceso de comunicación (ruta), pudiéndose utilizar varios de los enlaces previamente definidos, con la finalidad de simular las funciones de los enrutadores, los cuales permiten establecer varias vías para el envío de información. Con esto se modelan las posibles topologías de red de interconexión WAN.

A continuación se presenta una plantilla que indica el formato de este archivo.

```
<platform_description version="1">
  <cpu name="nombre-de-computador-o-host" power="capacidad-
    del-cpu-en-flops"/>
  <network_link name="nombre-o-numero-de-enlace"
    bandwidth="ancho-de-banda-en-bytes" latency="latencia-
    promedio-en-segundos"/>
  <route src="nombre-computador-o-host-origen-de-enlace"
    dst="nombre-computador-o-host-destino-de-
    enlace"><route_element name="nombre-o-número-de-enlace-
    1"/>…<route_element name="nombre-o-número-de-enlace-
    n"/>
   </route>
```

4.2. EJEMPLO NUMÉRICO

Para los escenarios experimentales se describen cuales son las combinaciones lógicas necesarias e interesantes para los distintos valores posibles de las variables y parámetros, configurando cada uno de los escenarios utilizados en cada ejecución de la simulación.

Con relación a la simulación del metaplanificador propuesto, la primera pregunta a responder es ¿cuál es la información que se requiere del experimento?, para responder a ésta, inicialmente se debe pensar en el hecho que se está proponiendo un protocolo para plataformas GRID, por ende, se debe obtener información sobre aspectos relacionados a su eficiencia, se debe saber si el protocolo es correcto, es decir, si hace lo que se supone que debe hacer. Además, es importante conocer cual es el valor agregado a nivel de rendimiento que la propuesta ofrece, es decir, en comparación a otros protocolos propuestos, qué ventajas o beneficios adicionales ofrece. También es necesario determinar la facilidad de implantación en las plataformas reales, y el beneficio que obtienen los usuarios y dueños de recursos, que son los entes físicos que poseen la necesidad real. Sin embargo, cuando se trata de las GRIDs, se debe considerar que el sistema de interconexión es Internet. En otras palabras, para simular el protocolo metaplanificador, en una primera instancia se debe establecer la caracterización de la red de interconexión sobre el cual se implementará, para este caso se debe caracterizar a Internet, y modelar su comportamiento es una labor que se escapa del alcance de este trabajo.

Para este trabajo se utilizó como marco de interconexión la red utilizada en el proyecto Reacción 2, desarrollada en el CENIT (anteriormente CNTI) a través del Comité Educativo de Reacciun, esto motivado a que se tiene planteado instalar una GRID, denominada GRID Venezuela, sobre esta red de interconexión. El esquema de interconexión propuesto se describe en [19], el cual se puede ver en la figura 11 (imagen extraída de [19]). En esta red, para su fase inicial, se interconectaron ocho (8) centros universitarios y de investigación: ULA, UCV, UC, USB, IVIC, UPEL, UDO y UCLA; a través de una red WAN, utilizando enlaces tipo E1 (34 Mbps), hacia un centro común (el CENIT), en otras palabras, la topología utilizada es una red WAN en estrella simple, teniendo como centro al CENIT.

Con lo que respecta al modelado de las aplicaciones en la GRID, dado que no se consiguieron datos para poder modelar y simular una aplicación específica dentro del marco de referencia de los escenarios GRID, y además considerando que al utilizar datos de una aplicación específica y simularla con SimGrid, se pierde generalidad en las pruebas ya que estarían solamente relacionadas a esa aplicación; se utilizó un modelo propuesto en [6] para la evaluación del rendimiento del protocolo utilizando métodos estocásticos, el cual se describe a continuación: Sea *D* un conjunto de dominios o redes

locales (por ejemplo los campus universitarios de Reacciun 2) presentes en una plataforma de múltiples sitios. Un dominio *Di* se conecta a una red global WAN a través de un switch, tal como se observa en la figura 12, donde *bwtd* es el ancho de banda de la conexión global y *latend* representa la latencia global para ese mismo enlace.

La carga de trabajo global corresponde a las tareas introducidas en el sistema global que están destinadas a las negociaciones del metaplanificador. Una tarea m_k posee los siguientes *atributos={tk, longk, prock, clientek}*, donde *tk* es el tiempo de entrada de la tarea al sistema global, *longk* es el requerimiento de computo (unidades de cómputo, por unidad de tiempo), *tipok* es el tipo de recurso que está solicitando (si es que la tarea se asocia a una solicitud de un corredor, en caso contrario no tendrá un tipo explicito asociado), y *clientek* es la organización o cliente que introdujo la tarea.

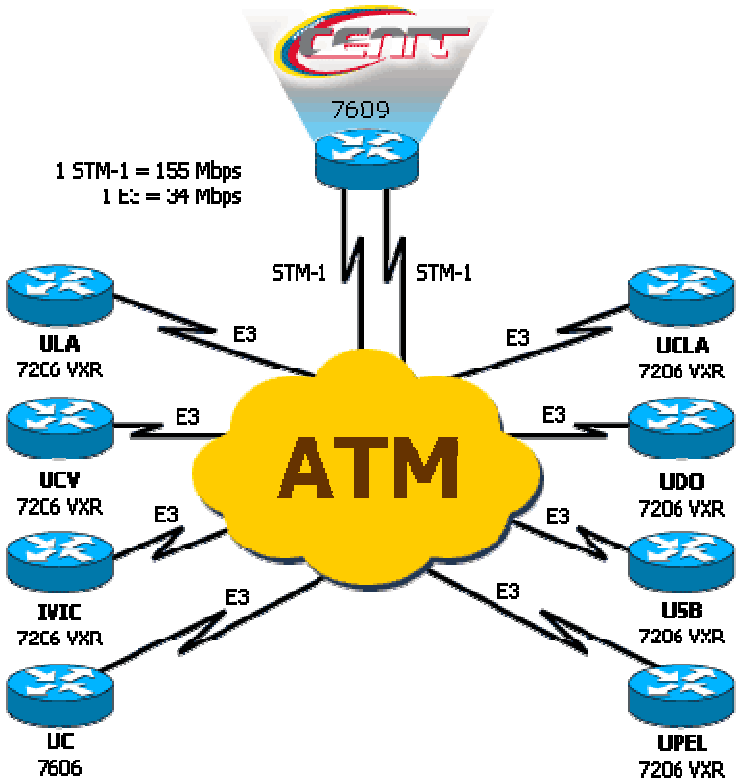


Figura 11. Modelo de Topología de Interconexión para la GRID

Las tareas son de control, es decir, para la realización de las negociaciones, que finalizarán en la ejecución de la solicitud emitida por el corredor en los recursos contratados, pero esta fase no se incluye en estas pruebas.

Según [6], los métodos para generar cargas de trabajo en los simuladores son: con aleatoriedad, con trazas derivadas de sistemas reales, y a través de métodos estocásticos. Los dos primeros no se pueden utilizar para realizar pruebas verdaderamente realistas, porque existen muchos tipos de plataformas y aplicaciones con requerimientos muy específicos, y sólo a través de métodos estocásticos se puede dar un enfoque general que represente en cierta medida (aproximaciones probabilísticas) los distintos casos. Por este motivo, es el método utilizado en este trabajo, es decir, para el modelado de las aplicaciones se utilizaron técnicas estocásticas; específicamente, la generación de tareas se realizó utilizando una distribución exponencial para determinar los tiempos entre cada generación, por ende, el número de llegadas sigue una distribución de Poisson. Para la distribución exponencial se utilizó como media 10 unidades de tiempo, lo cual es una representación arbitraria, que puede ser sustituida por un valor específico si se necesita modelar una aplicación en particular, lo cual no fue considerado en este trabajo, para poder mantener el enfoque genérico de las pruebas.

En este sentido, se define a A(T) como la función de distribución (exponencial) que genera el conjunto de los tiempos de entrada para un conjunto de tareas y sea λt (10) la tasa promedio de entradas, donde T es el conjunto de tareas que se generan.

Dado que este trabajo propone un metaplanificador, no se modelaron planificadores locales (para clusters), ni su estructura interna. Tampoco se modeló ejecución de las solicitudes en los recursos finales. Sólo se consideran los procesos asociados a la metaplanificación, y las tareas que se manejan están relacionadas a los procesos de control del protocolo, el cual finaliza cuando se realiza la preselección de los recursos, en otras palabras, cuando en el modelo se hace referencia a un tarea, dicha tarea debe estar asociada a una parte del proceso de negociación, y no a la utilización de un recurso en particular. En estos términos, cada dominio representa a una organización universitaria planteada en Reacción 2. Se establece que cada dominio ofrecerá recursos en la GRID Venezuela, por ende, tendrá un manejador de recursos asignado perteneciente a cada dominio, y además como las solicitudes de recursos en la GRID de Reacciun 2 también se generarán desde estos organismos, dichos dominios poseerán

también un corredor. En otras palabras, existirá una manejador y un corredor por cada institución conectada (por cada dominio), y ambos agentes se comunicarán por medio del mismo enlace de interconexión hacia el CENIT, donde se encontrará el agente coordinador, por medio del cual se conectará a las demás universidades.

En cuanto a las métricas y medidas utilizadas, se consideraron las tareas de solicitudes emanadas por los corredores para establecer los criterios de evaluación. Para una tarea del corredor *mk*, se definen dos estados: ejecución y finalizada. Si una tarea está en ejecución implica que ha sido generada por el corredor, y ha sido recibida por el coordinador, pero no se le ha dado respuesta por parte de éste, en caso de haber dado una respuesta, la tarea se considera como finalizada. Los instantes del inicio de cada estado se definen como: *ek* para el tiempo de inicio de la ejecución, y *fk* para el tiempo de finalización. Como se mencionó anteriormente el tiempo de generación de la tarea es *tk*.

De esta manera se define el tiempo de espera de una tarea como *ek-tk*. El tiempo de ejecución es *fk-ek*. El tiempo promedio de espera es:

$$t-espera = \frac{1}{cardinalidad(T)} \sum_{k \in T} ek - tk$$

El tiempo promedio de ejecución es:

$$t-ejecucion = \frac{1}{cardinalidad(T)} \sum_{k \in T} fk - ek$$

Se define como el tiempo de global de ejecución, como la cantidad de tiempo utilizada para culminar el proceso de metaplanificación, en cada periodo de tiempo ejecutado.

$$t-global = \max_{k \in T}(fk) - \min_{k \in T}(tk)$$

En cuanto a la herramienta de simulación utilizada, como se comentó anteriormente, en SimGrid la red se modela por medio de tres elementos: (1) los enlaces individuales que se caracterizan por su ancho de banda (bytes por segundos), y por su latencia (segundos), (2) las rutas de conexión entre dos hosts o nodos que se representan utilizando uno o más enlaces individuales, modelando el comportamiento de los enrutadores (ver [27]), y (3) cada host o nodo se caracteriza por su capacidad de

cómputo (flops), sin incluir su capacidad de almacenamiento, ya que lo que se pretende ejecutar son las tareas asociadas a la gestión de la planificación, para lo cual no se necesitan grandes capacidades de almacenamiento. Los valores que se le asignen a estos parámetros configuran los escenarios de red en cada simulación.

Los parámetros asociados a cada agente se especifican según cada caso, para los agentes corredores se deben indicar: el nombre del coordinador, el número de solicitudes que va a realizar, la cantidad de bytes que va a consumir cuando se genera una tarea para esta solicitud, la cantidad en flops del cómputo requerido para dicha tarea. Los tipo de recursos solicitados se generan en forma aleatoria entre el rango de tipos de recursos pre-establecido, para las pruebas se establecieron 6 tipo de recursos, y los montos ofrecidos para cada solicitud hecha también se generan aleatoriamente, con el fin de darle el enfoque dinámico a la simulación. Para el agente coordinador se debe especificar el número de corredores, el número de manejadores, la lista de nombres de los corredores, y de igual forma para los manejadores, el tamaño en bytes de las tareas de control que genera el coordinador, y la cantidad en flops que consume cada tarea. Para los manejadores se generan aleatoriamente los tipos de recurso que atiende y los montos de los precios inicial y mínimo del recurso que representa, por otro lado debe introducirse como parámetros de entrada la cantidad en bytes del consumo de ancho de banda de las tareas de control que genera, y el consumo del CPU en flops de cada una de sus tareas. El modelo considera conocer el listado de corredores y manejadores previamente, ya que lo que se quiere probar son las capacidades de respuesta del protocolo ante distintos números de solicitudes enviadas por los corredores previamente definidos, y que serán atendidos por los correspondientes manejadores, en otras palabras, el número de corredores y manejadores son variables controladas, no son dinámicas, dado el tipo de mediciones planteadas.

Para la herramienta de simulación, se considera que cada corredor representa a un dominio en la red Reacciun 2, de igual forma para cada manejador. Como se muestra en la figura 12, cada dominio se enlaza con el centro de la estrella a través de un enlace compartido por su corredor y su manejador, el punto de enlace de cada dominio representa tanto al corredor como al manejador.

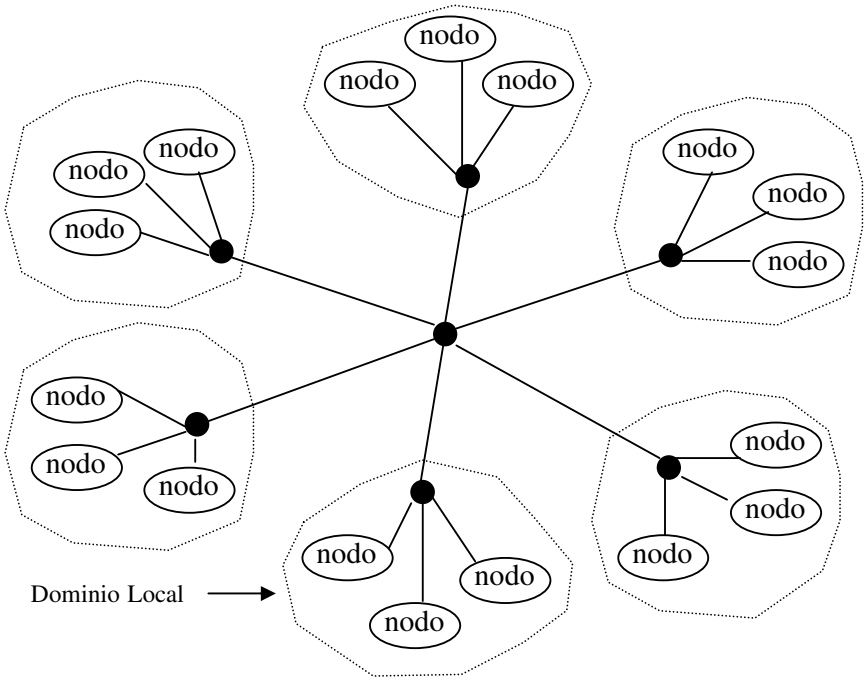


Figura 12. Esquema de Interconexión de dominios

En la tabla 1 se muestra el resumen de los datos utilizados para la simulación con respecto a cada nodo o dominio en la red:

Tabla 1. Datos del Modelo Simulado en SimGrid

A(T)	Distribución Exponencial (proceso Poisson)
λ(T)	10
Topología de Red	WAN tipo estrella simple
Ancho de Banda	34 Mbps (E1)
Latencia	Nula
Capacidad de CPU del Servidor Corredor y Manejador	Pentium IV de 2.2 GHz (2 operaciones Punto Flotante por Ciclo)
Capacidad en flops	4.4 Gflops
Número de Dominios	8
Número de Corredores por Dominio	1
Número de Manejadores por Dominio	1

En cuanto a las pruebas de correctitud del protocolo, no se hicieron debido a que las pruebas formales escapan al alcance de este trabajo, y representan parte del trabajo futuro recomendado.

4.2.1. Confiabilidad estadística de las pruebas:

Para poder conocer el nivel de errores manejados por los datos obtenidos, y dado que la generación de tareas se llevó a cabo por medio de una distribución exponencial, se determinó el grado de confiabilidad de los datos obtenidos de las simulaciones (tiempos de espera y de finalización de las tareas) generados a través del cálculo de intervalos de confianza, para poder tener un aproximado de los niveles de errores manejados. Para tal fin se utilizaron las fórmulas estadísticas indicadas en [18]. En este sentido, se generaron 30 corridas de simulación con el modelo implementado en SimGrid, y en cada corrida de simulación se generaron 100 solicitudes por cada dominio (centro universitario o de investigación), a través de cada uno de sus corredores, representando las solicitudes emanadas desde el interior de cada centro, dando un total de 800 solicitudes generadas por cada corrida. Para cada corrida de la simulación se midieron los tiempos de espera y los tiempos de finalización, tal como lo indicado en la sección anterior. En base a estas medidas se obtuvieron los promedios de todas las observaciones y se calcularon los intervalos de confianza. Dado el número de mediciones hechas, y según lo establecido en [18], la muestra obtenida se rige por la ley de los grandes números, por ende las observaciones se pueden considerar que siguen una distribución normal, y el cálculo de los intervalos de confianza se muestran a continuación:

Tabla 2. Intervalos de Confianza

Medición	Media Muestral	Intervalo de Confianza
Tiempo de Espera	**467.0579933**	**0.090645067**
Tiempo de finalización	**603.8980131**	**0.29397298**

4.2.2. Comparación Experimental:

Con la finalidad de comparar el protocolo propuesto con un protocolo con características muy básicas, se implementó un protocolo de metaplanificación centralizado con política de asignación FIFO, en el mismo simulador y se compararon sus métricas (tiempos de espera y de finalización) con los resultados obtenidos para el protocolo propuesto, en base a las 30 corridas del simulador. Las métricas consideradas tienen como objetivo conocer los tiempos de respuesta de ambos protocolos para poder comparar el comportamiento del protocolo propuesto, el cual incluye un esquema basado en modelos económicos de selección y asignación de recursos y usuarios mucho más ventajoso para los dueños de recursos y para los usuarios finales ya que incluye la capacidad de agregar elementos de calidad de servicio, y asignación según un nivel de confianza a través del esquema económico (mejores ofertas y menores precios), que el protocolo FIFO no puede ofrecer.

De esta forma, al igual que para el protocolo propuesto, se generaron 30 corridas para el metaplanificador FIFO, y se obtuvieron los tiempos de espera y de finalización de las tareas (100 tareas por dominio, dando un total de 800 tareas por corrida). Los resultados se observan en la tabla 3.

Tabla 3. Comparación entre el metaplanificador propuesto y el FIFO

	PROPUESTA		FIFO	
	Tiempos de espera (seg)	Tiempos de Finalización (seg)	Tiempos de espera (seg)	Tiempos de Finalización (seg)
Corrida 1	432.70434	504.26143	453.5416525	607.9725125
Corrida 2	487.2210963	601.651875	475.0529025	661.2349375
Corrida 3	452.8126438	520.2437625	500.51618	635.698675
Corrida 4	459.32981	744.1354125	471.9225875	679.35555
Corrida 5	470.782495	521.340625	498.14743	591.5797625
Corrida 6	453.5941912	557.15125	487.2413375	607.9222375
Corrida 7	460.4471975	624.12885	473.611495	591.417425
Corrida 8	460.54856	650.6033375	502.360245	670.667325
Corrida 9	512.3326438	666.1419	468.537745	612.3447375
Corrida 10	469.3801438	551.936625	482.2088375	657.01535
Corrida 11	474.5377925	589.4706625	456.4250875	550.9797625
Corrida 12	475.5926438	614.0242375	486.4875875	650.66685
Corrida 13	462.5790425	569.8879375	476.40993	629.3414
Corrida 14	468.4290425	572.7320875	455.580245	591.383925
Corrida 15	456.9966913	585.0533125	488.9763375	576.7824375
Corrida 16	475.0976438	634.6560375	476.4175875	628.9762125
Corrida 17	438.7754413	564.706825	463.2604025	600.565625
Corrida 18	438.1327925	810.188975	460.39368	560.7024625
Corrida 19	455.2679413	543.074825	431.520245	545.824375
Corrida 20	475.4951438	563.3019625	443.6391525	550.32135
Corrida 21	477.328745	604.2601125	492.2885225	641.969975
Corrida 22	481.73856	677.794125	472.777745	559.709225
Corrida 23	497.551245	594.48435	468.191495	660.7508125
Corrida 24	465.5277925	584.332125	488.73118	609.4158
Corrida 25	455.6091913	565.1667375	476.3050875	639.8627875
Corrida 26	481.8188938	591.1241	460.21493	577.3985
Corrida 27	434.8440425	644.52645	439.28252	555.9675375
Corrida 28	513.8848463	645.1924	449.6116525	616.4183875
Corrida 29	453.4677925	592.6477875	477.533995	642.2142125
Corrida 30	469.9113938	628.720275	496.661495	810.593425
Promedios	**467.0579933**	**603.8980131**	**472.461643**	**617.1684525**

4.2.3. Análisis de resultados:

En cuanto a los intervalos de confianza se puede decir que con una probabilidad del 98% los datos observados estarán dentro de ese rango (intervalo), o se puede decir que de cada 100 simulaciones que se generen con el modelo, en 98 de esas simulaciones los datos resultantes estarán dentro de esos intervalos de confianza. Lo que quiere decir que dado el estrecho tamaño del intervalo, las observaciones, son altamente confiables.

Dado que sólo se consideró un punto medición (100 solicitudes por cada corredor) dentro del espacio de posibles mediciones no se realizó un análisis comparativo entre

las medias muestrales obtenidas en las corridas de la simulación para ambos protocolos implementados en MetaSimGrid.

En cuanto a la comparación entre protocolos, se puede observar que no existen diferencias significativas entre los resultados de ambos protocolos (difieren entre el 1% y 2%). Esto se puede deber a que los tiempos de generación de tareas para ambos se hicieron a través de la misma distribución de probabilidad, y que los tiempos de procesamiento, tanto para las políticas FIFO como para las del protocolo propuesto, no difieren significativamente, ya que son tareas de control que requieren muy poco poder de cómputo y entre ambas no generaron diferencias significativas en las ejecuciones (mediciones de tiempos de espera y de finalización) globales del los protocolos. Esto implica, que al tener tiempos de respuestas iguales, el protocolo propuesto es mejor que el FIFO, ya que ofrece un conjunto de ventajas adicionales (selección de recursos y selección de usuarios con esquemas basados en modelos económicos), que permiten mejorar la calidad de la planificación, sin incurrir en consumo de tiempos adicionales.

Por otro lado, en las mediciones hechas, desde el punto de vista económico, la selección de recursos y de usuarios en el metaplanificador FIFO, no permitió escoger los mejores participantes en la GRID, y sólo se escogió en promedio al 30% de los participantes económicamente adecuados (mejores precios o mejores ofertas), en cada una de las corridas de la simulación; que en comparación al 100% obtenido con el protocolo propuesto, implica una ventaja evidente, en cuanto a la implementación de ambos. Para entender esto con mayor claridad, en la tabla 4 se muestran los datos relacionados a este tipo de selección, en donde se comparan, a través de algunos ejemplos, los resultados obtenidos en relación a la selección de los participantes económicamente adecuados. En particular se escogieron al azar 2 corridas de las simulaciones hechas sobre la propuesta y sobre el metaplanificador FIFO, y se puede observar que en el FIFO se escogieron recursos con montos muy altos en sus precios (para licitaciones), y usuarios con ofertas muy bajas (para las subastas).

Tabla 4. Datos Comparativos entre modelos

Corrida	Metaplanificador	Modelo Económico	Recurso o Usuario escogido	Precio u oferta seleccionada	Precios u ofertas adecuadas
2	Propuesta	Licitación	1	200	Menores a 300
2	FIFO	Licitación	1	900	Menores a 200
20	Propuesta	Subasta	5	700	Mayores a 650
20	FIFO	Subasta	5	10	Mayores a 500

En la tabla 4, se puede observar que el metaplanificador FIFO escogió usuarios y recursos fuera del rango aceptado desde el punto de vista económico.

El metaplanificador FIFO no permite conocer las características de los participantes en la GRID, ya que sólo utiliza como política de selección el orden de las llegadas, esto quiere decir que las contrataciones no consideran dichas características lo cual genera un riesgo adicional por tal desconocimiento. La propuesta incluye desarrollar la metaplanificación bajo el esquema de los modelos económicos, y esto implica que se deben conocer las características de los participantes, lo que le da un mayor grado de confiabilidad, y todo esto se logra obteniendo tiempos de respuestas similares.

La propuesta de este trabajo plantea un metaplanificador descentralizado, en la cual se pueden utilizar varios coordinadores cuando se necesita escalar a mayores volúmenes de requerimientos, por ejemplo, cuando en una GRID real existan aproximadamente 100.000 tareas o solicitudes que atender, la mejor forma de utilizar este protocolo sería estableciendo un conjunto de coordinadores que atiendan esta cantidad de requerimientos. Sin embargo, el protocolo FIFO, centralizado, no posee este tipo de versatilidad, ya que no es escalable, por poseer colas centralizadas para el manejo de tareas, lo que le impediría manejar volúmenes cercanos a las 100.000 solicitudes. En otras palabras, el protocolo FIFO no es escalable, dado su estructura centralizada, y por el contrario, el protocolo propuesto, al poder ser implementado con varios coordinadores, puede escalar a mayores niveles de exigencias. En este sentido, según las pruebas realizadas, los tiempos de respuestas fueron similares entre ambas metaplanificadores, pero se debe considerar que dichos tiempos se ajustan a un modelo donde se utilizó a un solo coordinador en el protocolo propuesto, para fines de evaluación, por lo cual se comportó parecido que el metaplanificador FIFO.

CONCLUSIÓN

Se diseñó e implemento un protocolo para la Metaplanificación en plataformas GRID, basado en los protocolos de Interacción de Sistemas Multiagentes establecidos en los estándares FIPA. El protocolo propuesto se diversificó en tres enfoques basados en el paradigma de los modelos económicos humanos: subastas, licitaciones y su combinación dinámica. De esta forma, en el proceso de metaplanificación, la selección de recursos y/o usuarios se determina por las estructuras de estos enfoques económicos.

Para la representación de dichos enfoques se utilizaron las técnicas de representación propuestas en AUML, y en particular, se utilizaron los diagramas de secuencias extendidos y los diagramas de colaboración, ambos pertenecientes a la familia de los diagramas de interacción.

Para la fase de implementación y evaluación de la propuesta se utilizaron las metodologías planteadas para el modelado y la simulación, y en particular se utilizó una herramienta para la simulación de planificadores de plataformas distribuidas denominada SimGrid, la cual es una librería escrita en lenguaje C.

El modelo representado en SimGrid, fue una GRID que utiliza una red de interconexión establecida en Venezuela para las comunicaciones de alta velocidad entre 8 universidades y centros de investigación (Reacciun 2), cada una representando un dominio. Esta red se estableció como patrón de interconexión para el modelo, y para la representación de las aplicaciones GRID, se utilizaron los métodos estocásticos, en particular los procesos Poisson, donde las tareas se generaron siguiendo una distribución exponencial, desde cada uno de los dominios (las 8 universidades y centros de investigación). Se generaron 30 simulaciones (corridas) con el modelo, y en cada una se generaron 100 tareas por cada dominio, dando un total de 800 tareas en cada simulación. Las métricas utilizadas fueron los tiempos de espera de las tareas y los tiempos de finalización. En un enfoque estadístico, se calcularon los intervalos de confianza para los tiempos promedios de espera y de finalización, dando un margen de error muy pequeño para las observaciones hechas. Por otro lado, se implementó en SimGrid un Metaplanificador centralizado con políticas FIFO, y se desarrolló el mismo esquema de simulaciones que para el protocolo propuesto. Esto se hizo con la intención

de comparar ambos protocolos, y como resultado se obtuvo que las diferencias entre los tiempos de respuesta en ambos no son significativas dado que las tareas de control para ambos protocolos requieren capacidades de cómputo muy similares, esto quiere decir que el metaplanificador propuesto es mejor, al ofrecer una gran cantidad de ventajas adicionales con respecto al FIFO y poseer tiempos de respuesta similares. El metaplanificador FIFO no permite conocer las características de los usuarios ante los dueños de recursos, y viceversa, por tal motivo no permite hacer una planificación basada en el conocimiento de las partes, ya que sólo considera el orden de llegada, y esto representa niveles de riesgos mayores en cuanto a la generación de las contrataciones, lo cual se evita con el protocolo propuesto, con tiempos de respuesta similares. Además, el protocolo propuesto permite atender mayores niveles de exigencia al poseer la cualidad de descentralización, lo que por el contrario, el metaplanificador FIFO no puede tener esos niveles de escalabilidad por tener una estructura centralizada, lo que seguramente implicará diferencias significativas en los tiempos de respuesta, si se utilizan mayores volúmenes de solicitudes.

REFERENCIAS BIBLIOGRÁFICAS

[1] Aguilar J.: "Grid Computing and Scheduling". Universidad de Los Andes (2005).

[2] Aida K., Takefusa A., Nakada H., Matsuoka S., Sekiguchi S. y Nagashima U.: "Performance Evaluation Model for Scheduling in Global Computing Systems". International Journal of High Performance Computing Applications. Vol. 14. (2000). 268-279.

[3] Bubendorfer K.: "Improving Resource Utilisation in Market Oriented Grid Management and Scheduling". ACM International Conference Proceeding Series. Vol. 167. (2006). 25-31.

[4] Buyya R., Abramson D. y Giddy J.: "A Case for Economy Grid Architecture for Service Oriented Grid Computing". 10th IEEE International Heterogeneous Computing Workshop in conjunction with IPDPS. (2001).

[5] Campo M.: "Tecnologías Middleware para el Desarrollo de Servicios en Entornos de Computación Ubicua". Tesis Doctoral. Universidad Carlos III de Madrid. (2004).

[6] Caron E., Garonne V. y Tsaregorodtsev A.: "Evaluation of Meta-scheduler Architectures and Task Assignment Policies for High Throughput Computing". Rapport De Recherche Inria. (2006).

[7] Legrand A., Marchal L. y Casanova H.: "Scheduling Distributed Applications: the SimGrid Simulation Framework". 3rd International Symposium on Cluster Computing and the Grid. (2003). 138.

[8] Casanova H.: "Simgrid: a Toolkit for the Simulation of Application Scheduling". Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid CCGrid. (2001).

[9] Chien C., Chang P. y Soo V.: "Market-Oriented Multiple Resource Scheduling in Grid Computing Environments". 19th International Conference on Advanced Information Networking and Applications. AINA papers, Vol.1, (2005) 867-872.

[10] Ernemann C., Hamscher V. y Yahyapour R.: "Economic Scheduling in Grid Computing". In Job Scheduling Strategies for Parallel Processing, Edinburgh, Scotland, (2002).

[11] Ernemann C., Hamscher V., Yahyapour R. y Streit A.: "Enhanced Algorithms for Multi-Site Scheduling". In 3rd Int'l Workshop on Grid Computing. (2002). 219-231.

[12] FIPA en línea: http://www.fipa.org. Consultado noviembre 2006.

[13] Foster I.: "Globus Toolkit Version 4: Software for Service-Oriented Systems".

[14] Foster I., Kesselman C. y Tuecke S.: "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". IFIP International Conference on Network and Parallel Computing. Springer-Verlag LNCS 3779. (2006). 2-13.

[15] Gómez J. y Mestras J.: "Análisis y Diseño de Sistemas Multi-Agentes". Curso de Doctorado. Universidad Complutense de Madrid, (2001-2002).

[16] Gradwell P.: "Distributed Combinatorial Resource Scheduling". AAMAS. (2005).

[17] Gradwell P.: "Grid Scheduling with Agents". 2nd International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS. (2003) 14-18.

[18] Mendenhall W., Scheaffer R. y Wackerly D.: "Estadística Matemática con Aplicaciones". Grupo Editorial Iberoamérica. México D. F. (1986). 503-524.

[19] REACCIUN2 en línea: http://www.reacciun2.edu.ve . Consultado en junio 2007.

[20] SIMGRID en línea: http://simgrid.gforge.inria.fr. Consultado en enero 2007.

[21] TERRA en línea: http://inmobiliaria.terra.es/ATREA/public/UCAGlosario.jsp. Consultado en abril 2007.

[22] Varela C., Ciancarini P. y Taura K.: "Worldwide Computing: Adaptive Middleware and Programming Technology for Dynamic Grid Environments". Scientific Programming Journal Special Issue on Dynamic Grids and Worldwide Computing. Vol. 13. IOS Press. (2005).

[23] WIKIPEDIA en línea: es.wikipedia.org/wiki/Trueque. Consultado en junio 2007.

[24] Wisnesky R.: "Evaluating Scheduling Algoritms on Distributed Computational Grids". In Proceedings of the 11th IEEE Symposium on HighPerformance Distributed Computing. (2002).

[25] Wu R., Chien A., Hiltunen M., Schlichting R. y Sen S.: "A High Performance Configurable Transport Protocol for Grid Computing". CCGRID Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid. Vol. 2. (2005). 1117-1125.

[26] Odell J., Van-Dyke H., Bauer B.: "Representing Agent Interaction Protocols in UML". Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, (2001). 121–140.

[27] Casanova H., Marchal L.: "A Network Model for simulation of Grid Application". Ecole Normale Superieure de Lyon, (2002).

ANEXO 1
Código del Simulador en SimGrid (lenguaje C)

```
/* Este programa en C representa un modelo bajo el esquema de SimGrid,
el cual es un Simulador para aplicaciones distribuidas heterogéneas.
Específicamente modela un Metaplanificador basado en sistemas
multiagentes
bajo los estándares de FIPA
Fecha: Marzo 2007
Autor: Rodolfo Sumoza, Universidad Simon Bolivar
*/

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include "msg/msg.h"
#include "xbt.h"
#include "metaplanificador.h"
//#define RAND_MAX 32000

/* Esta funcion es utilizada para que se puedan chequear que
cada uno de los procesos ha recibido los argumentos de forma correcta.
 */
static void print_args(int argc, char** argv)
{
  int i ;
  fprintf(stderr,"<");
  for(i=0; i<argc; i++)
    fprintf(stderr,"%s ",argv[i]);
  fprintf(stderr,">\n");
}


/**
 * Representacion de un cliente, a travÃ©s de su corredor
 **/
int corredor(int argc, char *argv[]) {
  m_host_t *coordinador = NULL;
  m_task_t *solicitudes = NULL;
  psolicitud_corredor_coordinador contenido, *pcontenido;
  prespuesta_coordinador_corredor contenidoR;
  int a;
  int numero_de_solicitudes = 0;
  double tam_solicitud_comp = 0;
  double tam_solicitud_comm = 0;
  double oferta_ini = 0;
  double oferta_min = 0;
  double oferta_max = 0;
  int tipo_solicitud;
  int i,o=0,oo;
  print_args(argc,argv);
  xbt_assert1(sscanf(argv[2],"%d", &numero_de_solicitudes),"Argumento
Invalido %s\n",argv[2]);
  xbt_assert1(sscanf(argv[3],"%lg", &tam_solicitud_comp),"Argumento
Invalido %s\n",argv[3]);
  xbt_assert1(sscanf(argv[4],"%lg", &tam_solicitud_comm),"Argumento
Invalido %s\n",argv[4]);

   /* Este parte se crean y envÃa las solicitudes al coordinador */
```

```
  coordinador = calloc(1,sizeof(m_host_t));
  coordinador[0] = MSG_get_host_by_name(argv[1]);
  if(coordinador == NULL) {
        INFO1("MÃ¡quina %s no definida. Se para la simulaciÃ³n! ",
argv[1]);
        abort();
  }
   char tareas[10000];
   solicitudes = calloc(numero_de_solicitudes,sizeof(m_task_t));
   pcontenido =
calloc(numero_de_solicitudes,sizeof(psolicitud_corredor_coordinador));
   srand(time(NULL));
   for (i = 0; i < numero_de_solicitudes; i++) {
         sprintf(tareas,"%d",i+1);
         contenido =
(psolicitud_corredor_coordinador)malloc(sizeof(solicitud_corredor_coor
dinador));
         oo = rand();
         tipo_solicitud = 1+(int)(6.0*rand()/(RAND_MAX+1.0));
         contenido->tipo = tipo_solicitud;
         oferta_ini = 1.0+1000.0*(rand()/(RAND_MAX+1.0));
         contenido->oferta_ini = oferta_ini;
         oferta_min = 1.0+1000.0*(rand()/(RAND_MAX+1.0));
         if (oferta_min > oferta_ini) oferta_min = oferta_ini;
         contenido->oferta_min = oferta_min;
         oferta_max = 1.0+1000.0*(rand()/(RAND_MAX+1.0));
         if (oferta_max < oferta_ini) oferta_max = oferta_ini;
         contenido->oferta_max = oferta_max;
         pcontenido[i] = contenido;
         o = (int)(10.0*log((1.0/(1.0-(rand()/(RAND_MAX+1.0))))));
         contenido->tarea = i+1;
         MSG_process_sleep(o);
         solicitudes[i] = MSG_task_create(tareas, tam_solicitud_comp,
tam_solicitud_comm, pcontenido[i]);
         INFO2("Creacion: %s %d", MSG_host_get_name(MSG_host_self()),
i+1);
         VERB2("Creacion: %s %d", MSG_host_get_name(MSG_host_self()),
i+1);
         printf("\n Creacion: %s %d %g",
MSG_host_get_name(MSG_host_self()), i+1, MSG_get_clock());

   }
   for (i = 0; i < numero_de_solicitudes; i++) {
         MSG_task_put(solicitudes[i],coordinador[0],PUERTO_corr_coor);
   }
  /*  Esta parte recibe las respuestas de de los coordinadores y le
envÃa su respuesta */
  int contador = numero_de_solicitudes;
  m_task_t respuesta;
  while(1) {
         respuesta = NULL;
         a = MSG_task_get(&respuesta, PUERTO_coor_corr);
         if (a == MSG_OK) {
           contenidoR = (prespuesta_coordinador_corredor)
MSG_task_get_data(respuesta);
           INFO2("Fin: %s
%d",MSG_host_get_name(MSG_host_self()),contenidoR->tarea);
           VERB2("Fin: %s
%d",MSG_host_get_name(MSG_host_self()),contenidoR->tarea);
```

```
           printf("\n Fin: %s %d
%g",MSG_host_get_name(MSG_host_self()),contenidoR-
>tarea,MSG_get_clock());
           free(contenidoR);
           MSG_task_execute(respuesta);
           MSG_task_destroy(respuesta);
           contador--;
           if (contador == 0) break;
         } else {
           INFO0("No se pudo obtener adecuadamente una respuesta");
           xbt_assert0(0,"Comportamiento inesperado");
         }
  }
  free(solicitudes);
  free(coordinador);
  free(pcontenido);
  return 0;
 } /* fin del corredor */



/**
 * Representacion del Coordinador.
 **/

int coordinador(int argc, char *argv[]) {

/**
Los parÃ¡metros pasados al coordinador son los siguientes: 1=Numero de
Corredores (clientes),
2=NÃºmero de Manejadores de recursos,3=NÃºmero de buscadores, apartir
del 4 argumento dependerÃ¡ de
la cantidad de participantes antes definidos, y se colocarÃ¡n en ese
mismo orden
**/

  int contador_corredor = 0;
  int contador_manejador = 0;
  double tam_solicitud_comp = 0;
  double tam_solicitud_comm = 0;
  m_host_t *corredores = NULL;
  m_host_t *manejadores = NULL;
  m_task_t respuesta_del_manejador = NULL;
  m_task_t solicitud_al_manejador = NULL;
  m_task_t respuesta_al_corredor = NULL;
  m_task_t respuesta_al_manejador = NULL;
  plista_solicitudes_manejadores lista_general = NULL, aux_1g = NULL,
aux_2g = NULL;
  plista_manejador aux_1m = NULL, aux_2m = NULL, aux_3m = NULL;
  plista_solicitud aux_1s = NULL, aux_2s = NULL, aux_3s = NULL,
contenido1 = NULL, contenido2 = NULL;
  psolicitud_coordinador_manejador contenido1_cm = NULL;
  precursos_manejador aux_1m = NULL;
  prespuesta_coordinador_manejador respuesta_cm = NULL;
  prespuesta_coordinador_corredor respuesta_cc = NULL;
  int i,a,tipo;
  print_args(argc,argv);
  xbt_assert1(sscanf(argv[1],"%d", &contador_corredor),"Argumento
InvÃ¡lido %s\n",argv[1]);
  xbt_assert1(sscanf(argv[2],"%d", &contador_manejador),"Argumento
InvÃ¡lido %s\n",argv[2]);
```

```
  xbt_assert1(sscanf(argv[3],"%lg", &tam_solicitud_comp),"Argumento
InvÃ¡lido %s\n",argv[3]);
  xbt_assert1(sscanf(argv[4],"%lg", &tam_solicitud_comm),"Argumento
InvÃ¡lido %s\n",argv[4]);
  manejadores = calloc(contador_manejador, sizeof(m_host_t));
  corredores = calloc(contador_corredor, sizeof(m_host_t));
  for (i = 0; i < contador_manejador; i++) {
      manejadores[i] =
MSG_get_host_by_name(argv[5+contador_corredor+i]);
      corredores[i] = MSG_get_host_by_name(argv[5+i]);
      if(manejadores[i]==NULL) {
        INFO1("MÃ¡quina-Manejador %s no definida. Se para la
simulaciÃ³n! ", argv[5+contador_corredor+i]);
        abort();
      }
  }
  int conta = 800;
                  /* OrganizaciÃ³n de los Procesos */
  do {
    /* Esta parte recibe las solicitudes desde los corredores, las
regitra en la lista y les envias a los manejadores encontrados la
solicitud de ofertyas. Esata parte incluye el proceso de busqueda
(esta implicito) */
       m_task_t solicitud_corredor = NULL;
       a = MSG_task_get(&solicitud_corredor,PUERTO_corr_coor);
       if (a == MSG_OK) {
          conta--;
          INFO2("Proceso: %s
%d",MSG_host_get_name(MSG_task_get_source(solicitud_corredor)),((psoli
citud_corredor_coordinador) MSG_task_get_data(solicitud_corredor))-
>tarea);
          printf("\n Proceso: %s %d
%g",MSG_host_get_name(MSG_task_get_source(solicitud_corredor)),((psoli
citud_corredor_coordinador) MSG_task_get_data(solicitud_corredor))-
>tarea,MSG_get_clock());
          VERB2("/t Proceso: %s
%d",MSG_host_get_name(MSG_task_get_source(solicitud_corredor)),((psoli
citud_corredor_coordinador) MSG_task_get_data(solicitud_corredor))-
>tarea);
          tipo = ((psolicitud_corredor_coordinador)
MSG_task_get_data(solicitud_corredor))->tipo;
          if (lista_general==NULL){
                lista_general = (plista_solicitudes_manejadores)
malloc(sizeof(lista_solicitudes_manejadores));
                lista_general->lista_m = NULL;
                lista_general->lista_s = NULL;
                lista_general->sig = NULL;
                lista_general->tipo = tipo;
                lista_general->contador_m = 0;
                lista_general->contador_s = 0;
                aux_1g = lista_general;
          } else {
                aux_2g = lista_general;
                while (aux_2g != NULL) {
                     aux_1g = aux_2g;
                     if (aux_1g->tipo == tipo)
                         break;
                     else {
                          aux_2g = aux_1g->sig;
                          if (aux_2g == NULL) {
```

```
                                aux_2g =
(plista_solicitudes_manejadores)
malloc(sizeof(lista_solicitudes_manejadores));
                                aux_2g->lista_s = NULL;
                                aux_2g->lista_m = NULL;
                                aux_2g->sig = NULL;
                                aux_2g->tipo = tipo;
                                aux_2g->contador_m = 0;
                                aux_2g->contador_s = 0;
                                aux_1g->sig = aux_2g;
                           }
                      }
                 }
          }
          aux_1s = (plista_solicitud) malloc(sizeof(lista_solicitud));
          aux_1s->tipo = ((psolicitud_corredor_coordinador)
MSG_task_get_data(solicitud_corredor))->tipo;
          aux_1s->oferta_ini = ((psolicitud_corredor_coordinador)
MSG_task_get_data(solicitud_corredor))->oferta_ini;
          aux_1s->oferta_min = ((psolicitud_corredor_coordinador)
MSG_task_get_data(solicitud_corredor))->oferta_min;
          aux_1s->oferta_max = ((psolicitud_corredor_coordinador)
MSG_task_get_data(solicitud_corredor))->oferta_max;
          aux_1s->tarea = ((psolicitud_corredor_coordinador)
MSG_task_get_data(solicitud_corredor))->tarea;
          aux_1s->corredor = (char *)
MSG_host_get_name(MSG_task_get_source(solicitud_corredor));
          aux_1s->sig = NULL;
          aux_1g->contador_s++;
          if (aux_1g->lista_s == NULL)
                   aux_1g->lista_s = aux_1s;
          else {
                   contenido1 = aux_1g->lista_s;
                   contenido2 = aux_1s;
                   if (contenido1->oferta_max < contenido2-
>oferta_max) {
                         aux_1s->sig = aux_1g->lista_s;
                         aux_1g->lista_s = aux_1s;
                   } else {
                         aux_2s = aux_1g->lista_s;
                         aux_3s = aux_1g->lista_s->sig;
                         while (aux_3s != NULL) {
                              if ((aux_1s->oferta_max) < (aux_3s-
>oferta_max)){
                                   aux_1s->sig = aux_3s;
                                   aux_2s->sig = aux_1s;
                                   break;
                              } else {
                                   aux_2s = aux_3s;
                                   aux_3s = aux_2s->sig;
                              }
                         }
                         if (aux_3s == NULL) aux_2s->sig = aux_1s;
                    }
          }
          MSG_task_execute(solicitud_corredor);
          MSG_task_destroy(solicitud_corredor);
       }
  }while(conta>0);
```

```
  aux_1g = lista_general;
  while (aux_1g!=NULL) {
         for (i=0; i < contador_manejador; i++){
              solicitud_al_manejador = NULL;
              contenido1_cm = (psolicitud_coordinador_manejador)
malloc (sizeof(solicitud_coordinador_manejador));
              contenido1_cm->tipo = aux_1g->tipo;
              contenido1_cm->manejador = i;
              solicitud_al_manejador =
MSG_task_create("Solicitud",tam_solicitud_comp, tam_solicitud_comm,
contenido1_cm);
              MSG_task_put(solicitud_al_manejador, manejadores[i],
PUERTO_coor_man);
              respuesta_del_manejador = NULL;
              aux_lm = NULL;
              a = MSG_task_get(&(respuesta_del_manejador),
PUERTO_man_coor);
              if (a == MSG_OK) {
                    aux_lm = ((prespuesta_manejador_coordinador)
MSG_task_get_data(respuesta_del_manejador))->lista;
                    while (aux_lm != NULL){
                        if (aux_1g->tipo == aux_lm->tipo) {
                           aux_1g->contador_m++;
                           aux_1m = (plista_manejador)
malloc(sizeof(lista_manejador));
                           aux_1m->precio_ini = aux_lm->precio_ini;
                           aux_1m->precio_min = aux_lm->precio_min;
                           aux_1m->manejador =
((prespuesta_manejador_coordinador)MSG_task_get_data(respuesta_del_man
ejador))->manejador;
                           aux_1m->sig = NULL;
                           if (aux_1g->lista_m != NULL) {
                                   if (aux_1g->lista_m->precio_min >
aux_1m->precio_min){
                                        aux_1m->sig = aux_1g->lista_m;
                                        aux_1g->lista_m = aux_1m;
                                   } else {
                                        aux_2m = aux_1g->lista_m;
                                        aux_3m = aux_2m->sig;
                                        while (aux_3m != NULL) {
                                             if (aux_3m->precio_min >
aux_1m->precio_min){
                                                  aux_1m->sig =
aux_3m;
                                                  aux_2m->sig =
aux_1m;
                                                  break;
                                             } else {
                                                  aux_2m = aux_3m;
                                                  aux_3m = aux_2m-
>sig;
                                             }
                                        }
                                        if (aux_3m == NULL) aux_2m-
>sig = aux_1m;
                                   }
                           } else {
                                  aux_1m->sig = NULL;
                                  aux_1g->lista_m = aux_1m;
                           }
```

```
                        }
                        aux_lm = aux_lm->sig;
                      }
                      MSG_task_execute(respuesta_del_manejador);
                      MSG_task_destroy(respuesta_del_manejador);
                }
          }
          aux_1g = aux_1g->sig;
  }


aux_1g = lista_general;
while (lista_general != NULL) {
            if (aux_1g->contador_m  < aux_1g->contador_s) {  //Caso
Subasta
                  aux_1s = aux_1g->lista_s;
                  aux_1m = aux_1g->lista_m;
                  while (aux_1g->contador_s > 0) {
                     if (aux_1m != NULL){
                        respuesta_cc =
(prespuesta_coordinador_corredor) malloc
(sizeof(respuesta_coordinador_corredor));
                        respuesta_cc->tipo = aux_1g->tipo;
                        respuesta_cc->tarea = aux_1s->tarea;
                        respuesta_cm =
(prespuesta_coordinador_manejador) malloc
(sizeof(respuesta_coordinador_manejador));
                        if (aux_1m->precio_min < aux_1s->oferta_max) {
                            respuesta_cc->manejador = (char *)
MSG_host_get_name(manejadores[aux_1m->manejador]);
                            respuesta_cc->precio = aux_1s->oferta_max;
                            respuesta_cm->corredor = aux_1s->corredor;
                            respuesta_cm->precio = aux_1s->oferta_max;
                        } else {
                            respuesta_cc->manejador = "NingunoA";
                            respuesta_cc->precio = 0;
                            respuesta_cm->corredor = "NingunoB";
                            respuesta_cm->precio = 0;
                        }
                        respuesta_cm->tipo = aux_1g->tipo;
                        respuesta_al_manejador = MSG_task_create
(NULL,tam_solicitud_comp, tam_solicitud_comm,respuesta_cm);
                        MSG_task_put
(respuesta_al_manejador,manejadores[aux_1m-
>manejador],PUERTO_coor_man);
                        aux_1g->lista_m = aux_1m->sig;
                        aux_1g->contador_m--;
                        free(aux_1m);
                        aux_1m = aux_1g->lista_m;
                     } else {
                        respuesta_cc =
(prespuesta_coordinador_corredor) malloc
(sizeof(respuesta_coordinador_corredor));
                        respuesta_cc->tipo = aux_1g->tipo;
                        respuesta_cc->manejador = "NingunoC";
                        respuesta_cc->precio = 0;
                        respuesta_cc->tarea = aux_1s->tarea;
                     }
                     respuesta_al_corredor = MSG_task_create
("Respuesta Coordinador-Corredor", tam_solicitud_comp,
tam_solicitud_comm,respuesta_cc);
```

```
                    MSG_task_put
(respuesta_al_corredor,MSG_get_host_by_name(aux_1s-
>corredor),PUERTO_coor_corr);
                    aux_1g->lista_s = aux_1s->sig;
                    aux_1g->contador_s--;
                    free(aux_1s);
                    aux_1s = aux_1g->lista_s;
                }
            } else { // Caso licitacion
                aux_1s = aux_1g->lista_s;
                aux_1m = aux_1g->lista_m;
                while (aux_1g->contador_m > 0) {
                    respuesta_cm = (prespuesta_coordinador_manejador)
malloc (sizeof(respuesta_coordinador_manejador));
                    respuesta_cm->tipo = aux_1g->tipo;
                    if (aux_1g->lista_s != NULL){
                        aux_1s = aux_1g->lista_s;
                        respuesta_cc =
(prespuesta_coordinador_corredor) malloc
(sizeof(respuesta_coordinador_corredor));
                        respuesta_cc->tipo = aux_1g->tipo;
                        respuesta_cc->tarea = aux_1s->tarea;
                        if (aux_1m->precio_min <= aux_1s->oferta_min){
                            respuesta_cc->manejador = (char
*)MSG_host_get_name(manejadores[aux_1m->manejador]);
                            respuesta_cc->precio = aux_1m->precio_min;
                            respuesta_cm->corredor = aux_1s->corredor;
                            respuesta_cm->precio = aux_1m->precio_min;
                        } else {
                            respuesta_cc->manejador = "NingunoD";
                            respuesta_cc->precio = 0;
                            respuesta_cm->corredor = "NingunoE";
                            respuesta_cm->precio = 0;
                        }
                        respuesta_al_corredor = MSG_task_create
("Respuesta Coordinador-Corredor", tam_solicitud_comp,
tam_solicitud_comm,respuesta_cc);
                        MSG_task_put
(respuesta_al_corredor,MSG_get_host_by_name(aux_1s-
>corredor),PUERTO_coor_corr);
                        aux_1g->lista_s = aux_1s->sig;
                        aux_1g->contador_s--;
                        free(aux_1s);
                    } else {
                        respuesta_cm->corredor = "NingunoF";
                        respuesta_cm->precio = 0;
                    }
                    respuesta_al_manejador = MSG_task_create (NULL,
tam_solicitud_comp, tam_solicitud_comm,respuesta_cm);
                    MSG_task_put
(respuesta_al_manejador,manejadores[aux_1m-
>manejador],PUERTO_coor_man);
                    aux_1g->lista_m = aux_1m->sig;
                    aux_1g->contador_m--;
                    free(aux_1m);
                    aux_1m = aux_1g->lista_m;
                }
            }
          lista_general = aux_1g->sig;
          free (aux_1g);
          aux_1g = lista_general;
```

```
    }

  return 0;
}  /* Fin del Coordinador */


/**
 * Representacion de un Manejador de recursos.
 **/

int manejador(int argc, char *argv[]) {
  m_task_t respuesta_al_coordinador;
  MSG_error_t a;
  int i, num_recursos;
  double tam_solicitud_comp;
  double tam_solicitud_comm;
  precursos_manejador recursos, ini_recursos = NULL, aux_r;
  prespuesta_manejador_coordinador respuesta_mc;
  print_args(argc, argv);
  xbt_assert1(sscanf(argv[1], "%d", &num_recursos), "Argumento
InvÃ¡lido '%s'.",argv[1]);
  xbt_assert1(sscanf(argv[2], "%lg", &tam_solicitud_comp), "Argumento
InvÃ¡lido '%s'.",argv[2]);
  xbt_assert1(sscanf(argv[3], "%lg", &tam_solicitud_comm), "Argumento
InvÃ¡lido '%s'.",argv[3]);
  srand(time(NULL));
  for(i=0; i < num_recursos; i++){
        recursos = (precursos_manejador) malloc
(sizeof(recursos_manejador));
        recursos->tipo = 1+(int)(6.0*rand()/(RAND_MAX+1.0));
        recursos->precio_ini = 1.0+1000.0*rand()/(RAND_MAX+1.0);
        recursos->precio_min = 1.0+1000.0*rand()/(RAND_MAX+1.0);
        recursos->sig = NULL;
        if (recursos->precio_ini < recursos->precio_min)  recursos-
>precio_min = recursos->precio_ini;
        if (ini_recursos == NULL) {
            ini_recursos = recursos;
        } else {
            aux_r = ini_recursos;
            while (aux_r->sig != NULL){
                aux_r = aux_r->sig;
            }
            aux_r->sig = recursos;
        }
  }
  while (1) {
       m_task_t solicitud_coordinador = NULL;
       a = MSG_task_get(&(solicitud_coordinador), PUERTO_coor_man);
       if (a == MSG_OK) {
          if (MSG_task_get_name(solicitud_coordinador)!=NULL) {
                  respuesta_mc = (prespuesta_manejador_coordinador)
malloc (sizeof(respuesta_manejador_coordinador));
                  respuesta_mc->lista = ini_recursos;
                  respuesta_mc->manejador =
((psolicitud_coordinador_manejador)MSG_task_get_data(solicitud_coordin
ador))->manejador;
                  respuesta_al_coordinador =
MSG_task_create("Respuesta del Manejador", tam_solicitud_comp,
tam_solicitud_comm,respuesta_mc);
                  MSG_task_put(respuesta_al_coordinador,
MSG_task_get_source(solicitud_coordinador),PUERTO_man_coor);
```

```
                MSG_task_execute(solicitud_coordinador);
                MSG_task_destroy(solicitud_coordinador);
          } else {
               //INFO1("Soy el manejador %s. Recibida del Coordinador
la informacion final",MSG_host_get_name(MSG_host_self()));
               MSG_task_execute(solicitud_coordinador);
               MSG_task_destroy(solicitud_coordinador);
          }
       } else {
       }
  }
  return 0;
}


void simulador(const char *archivo_plataforma,const char
*archivo_aplicacion)
{

  MSG_config("surf_workstation_model","KCCFLN05");
  {                              /*  ConfiguraciÃ³n de la SimulaciÃ³n
*/
    MSG_set_channel_number(CANAL_MAX);
    MSG_paje_output("msg_test.trace");
    MSG_create_environment(archivo_plataforma);
  }
  {                             /*   Desarrollo de la AplicaciÃ³n  */
    MSG_function_register("corredor", corredor);
    MSG_function_register("coordinador", coordinador);
    MSG_function_register("manejador", manejador);
    MSG_launch_application(archivo_aplicacion);
  }
  MSG_main();
  INFO1("Tiempo de la SimulaciÃ³n %g",MSG_get_clock());
} /* Fin de la SimulaciÃ³n */


int main(int argc, char *argv[])
{
  MSG_global_init_args(&argc,argv);
  if (argc < 3) {
      printf ("Deben utilizarse dos argumentos: %s
<archivo_plataforma> <archivo_aplicacion>.\n", argv[0]);
      printf ("ejemplo: %s msg_platform.xml msg_deployment.xml\n",
argv[0]);
      exit(1);
  }
  simulador(argv[1],argv[2]);
  MSG_clean();
  return (0);
} /* Fin del bloque principal */
```