Aumentando las Consultas de Usuario para Propósitos de Adaptación en Ambientes Nómadas

Augmenting User Queries for Adaptation Purposes in Nomadic Environments

Angela Carrillo Ramos, PhD.[1], Marlène Villanova-Oliver, PhD.[2],
Jérôme Gensel, PhD.[2], Hervé Martin, PhD.[2]
[1]Pontificia Universidad Javeriana, Bogotá, Colombia
[2]Laboratorio de Informática de Grenoble, Francia
angela.carrillo@javeriana.edu.co; {villanov, gensel, martin}imag.fr

Recibido para revisión 15 de Septiembre de 2007, Aceptado 30 de Noviembre de 2007, Versión final 4 de Diciembre de 2007

***Resumen*—PUMAS es un framework basado en agentes y una aproximación Punto a Punto (P2P), que permite al usuario nómada acceder diferentes fuentes de información a través de diversos tipos de dispositivo, eventualmente móviles. PUMAS provee al usuario nómada una información adaptada a sus preferencias y a las características de su contexto de uso (compuesto de características tales como localización, momento de conexión, características del dispositivo utilizado, entre otras). Con el fin de entregar los resultados adaptados, PUMAS dispone de mecanismos de enriquecimiento de la consulta inicial mediante la adición de criterios de adaptación que tienen en cuenta las preferencias del usuario y el contexto de uso. Esta fase de enriquecimiento de la consulta precede a un proceso de enrutamiento de consultas compuesto de tres actividades: el análisis de la consulta, la selección de las fuentes de información capaces de responderla y la redirección de la consulta hacia estas fuentes. Después de un estado del arte consagrado al enrutamiento de consultas en ambientes P2P, presentamos el framework PUMAS. Diferentes escenarios de uso de PUMAS son descritos, en particular los procesos de enriquecimiento y de enrutamiento de consultas que se ejecutan con el fin de proveer al usuario, una información adaptada a sus características.**

***Palabras Clave*—Sistemas P2P, Agentes Inteligentes, Enrutamiento de Consultas, PUMAS, Ambiente Nómada.**

***Abstract*—PUMAS is a framework based on agents and an approach Peer to Peer (P2P), which allows to nomadic user to access several sources of information through different types of devices, eventually mobile. PUMAS provides to nomadic user information adapted to her/his preferences and to the characteristics of the context of use (composed of characteristics such as location, connection moment, characteristic of the used device). In order to deliver adapted results, PUMAS has mechanisms of enrichment of the initial query by the addition of criteria accounting for the user preferences and the context of use. This phase of enrichment of the query precedes a process of query routing composed of three activities: the analysis of the query, the selection of the information sources which are able to answer it and the redirection of the query towards these sources. After a state of the art dedicated to the query routing in P2P environments, we present the framework PUMAS. Several scenarios of use of PUMAS are described, in particular, the processes of enrichment and query routing which PUMAS executes in order to deliver to the user information adapted to her/his characteristics.**

***Keywords*—P2P Systems, Agents, Query Routing, PUMAS, Nomadic Environment.**

I. Introducción

LOS dispositivos móviles (*DM*, *e.g.*, *PDA*, teléfonos, computadores portátiles, *etc.*) son utilizados hoy en día con el fin de acceder fuentes de información distantes al igual que sirven como fuentes de información. Así mismo, según sus características y capacidades físicas, el dispositivo no puede almacenar sino pequeñas cantidades de información (por ejemplo, algunos archivos). Los *DM* permiten el acceso, búsqueda, despliegue y almacenamiento de información. En este contexto, garantizar el acceso a numerosas fuentes de información y compartir recursos entre usuarios requiere hoy en día disponer de arquitecturas y de tecnologías que posean un gran potencial comunicativo. Este potencial es una de las características principales de los sistemas *Punto a Punto* (*P2P*).

El tratamiento de las consultas de usuarios nómades se ha convertido en un proceso cada vez más complejo, ya que el

resultado de las consultas puede provenir de diferentes fuentes de información. Además del número de fuentes potenciales de información, la heterogeneidad en su estructura y en sus modalidades de acceso puede ser un obstáculo para su explotación. Adicionalmente, es necesario considerar la entrega al usuario de la información mejor adaptada a sus características y a aquellas de su dispositivo de acceso. Parece entonces pertinente analizar las consultas con el fin de identificar las fuentes más apropiadas que puedan responderlas (lo que supone un conocimiento de las fuentes y de la información que manejan), además de evaluar y de integrar sus resultados. Estas actividades corresponden a un proceso tradicional de enrutamiento de consultas [20].

Sin embargo, estas actividades no tratan explícitamente de la adaptación de la información. Nuestra propuesta es una respuesta en este sentido y se basa en un enriquecimiento de la consulta, enriquecimiento que antecede el proceso de enrutamiento. Este enriquecimiento "*aumenta*" la consulta inicial adicionándole criterios de adaptación tales como las *preferencias del usuario* y características del *contexto de uso* de la sesión actual. En nuestro trabajo, el *contexto de uso* está compuesto de información sobre la localización del usuario, las características del *DM*, los derechos de acceso del usuario y sus actividades en el sistema [5].

Esta propuesta de enriquecimiento de la consulta se basa en el hecho de que diferentes factores pueden tener una influencia en la ejecución del enrutamiento (en el sentido en que las fuentes destinatarias de las consultas pueden cambiar):

- Los cambios de localización del usuario pueden producir cambios en términos de acceso y de necesidades de información [18]. Esto también es válido ante un cambio de dispositivo;
- Las preferencias de un usuario son dependientes del contexto [5] en el cual se ejecuta la consulta; todo cambio contextual puede traer consecuencias en la consulta, y por consiguiente en la selección de las fuentes a ser interrogadas;
- Las características y restricciones técnicas del *DM* del usuario nómada pueden producir problemas de despliegue de la información que serían deseables de anticipar.

Con el fin de tener en cuenta factores como los mencionados anteriormente, se propone *PUMAS* (*Peer Ubiquitous Multi-Agent System*) [3] un framework basado en agentes que provee a un usuario nómada una información adaptada a sus *preferencias* y a su *contexto de uso*, sin importar el dispositivo de acceso (móvil o no) utilizado. *PUMAS* ofrece igualmente los medios de interrogar diferentes fuentes de información; dichas fuentes son Sistemas de Información (*SI*) localizadas en servidores, o son simples archivos almacenados en un dispositivo móvil de tipo *PDA* por ejemplo. Los agentes de *PUMAS* implementan el mecanismo de enriquecimiento de la consulta inicial mediante la adición de criterios teniendo en cuenta las *preferencias del usuario* y el *contexto de uso*. Esta fase de enriquecimiento de la consulta antecede el proceso de enrutamiento.

En *PUMAS*, los usuarios se comunican entre ellos gracias a una arquitectura *P2P* híbrida. Se tomaron en cuenta para el modelo las recomendaciones de Shizuka *et al.* [16] quienes consideran que el uso de una arquitectura *P2P* Híbrida es apropiada para:

- la prevención de problemas de seguridad ligados a la movilidad de los agentes;
- la comunicación punto a punto o en "*broadcast*" entre los agentes;
- la gestión de los estados de los agentes (por ejemplo, "conectado", "desconectado", "suspendido", *etc.*) y de sus servicios ofrecidos.

PUMAS se compone de cuatro Sistemas Multi-Agentes (*SMA*) respectivamente encargados de la conexión, de la comunicación, de la gestión de las fuentes de información y de la adaptación de dicha información. De manera general, los *SMA* se asemejan a los sistemas *P2P* en numerosos puntos: Cada agente es un par (punto) en la medida en que él puede llevar a cabo sus tareas, independiente del servidor y de otros agentes. Los sistemas *P2P* [14] se caracterizan por: *i)* una comunicación directa entre pares sin comunicación a través de un servidor específico; *ii)* la autonomía de un par para llevar a cabo las tareas asignadas. Basado en una aproximación *P2P*, un *SMA* debe representar el conocimiento requerido por cada agente para llevar a cabo sus tareas asociadas a los diferentes roles que puede desempeñar (por ejemplo, cliente, servidor, coordinador) [12].

Este artículo está organizado de la siguiente manera: la sección 2 presenta los sistemas "*Peer to Peer*" (*P2P*) y sus características principales. La sección 3 define el proceso de enrutamiento de consultas en los sistemas *P2P* y se expone el estado del arte de este tema. Una visión general del framework *PUMAS* es presentada en la sección 4. Enseguida, la sección 5 se consagra a la presentación de los escenarios que ilustran el uso de *PUMAS*. En particular, describimos cómo una consulta es enriquecida para propósitos de adaptación, y presentamos el proceso de enrutamiento de consultas que se basa en el análisis de esta consulta y su redirección hacia las fuentes de información capaces de responderlas. Concluimos este artículo en la sección 6.

II. SISTEMAS "PEER TO PEER" (P2P)

Stuckenschmidt *et al.* [17] definen la informática *Punto a Punto* ("*Peer to Peer*", *P2P*) como una forma de computación distribuida que implica un gran número de nodos informáticos (los pares). Estos pares son autónomos y cooperativos, se comunican y comparten recursos y servicios disponibles, en ausencia de todo control central. En conjunto, constituyen sistemas altamente dinámicos: los nodos se unen o dejan el sistema y cambian tanto de contenidos como de

roles constantemente [16]. Por ejemplo, un par puede desempeñar el rol de *servidor* cuando provee datos, información y servicios; de *mediador* cuando analiza y encamina las necesidades de las consultas, y de *cliente* cuando accede a la información manejada por otros pares [22].

Stuckenschmidt *et al.* [17] distinguen tres categorías de *sistemas P2P* según el tipo de recursos compartidos:

- aplicaciones en las cuales los pares comparten recursos de cálculo (también llamadas "*Grid Computing*") ;
- aplicaciones en las cuales los pares comparten la lógica de aplicación (también llamadas "*Arquitecturas basadas en servicios*") ;
- aplicaciones en las cuales los pares comparten información (también llamados "*Sistemas P2P clásicos*").

El trabajo presentado aquí se sitúa en el marco de los s*istemas P2P* clásicos. Estos sistemas se dividen generalmente en dos grupos [10]: Los *sistemas P2P no estructurados* y los *sistemas P2P estructurados*.

Los ***sistemas P2P no estructurados*** comprenden los *sistemas* ***P2P Puros*** y los *sistemas* ***P2P Híbridos***. En un ***sistema P2P Puro***, todos los pares son iguales y no hay funcionalidad centralizada. Cada par posee una lista de pares con los cuales se puede comunicar, trabajar y compartir información. Sin embargo, si se considera necesario, es igualmente capaz de contactar otros pares (por ejemplo, consultando el anuario de los pares del sistema). En los *sistemas P2P Puros*, los mensajes son enviados sin la mediación de un servidor. En un ***sistema P2P Híbrido***, los pares están organizados en dos capas: la capa *superior* corresponde a los "*super-pares*" y la capa *inferior* corresponde a los "*pares comunes*". Cada par "*común*" puede conectarse a uno o varios *super-pares*. Los *super-pares* desempeñan el rol de servidor con el fin de facilitar la localización de recursos. Un par puede conectarse a un servidor indexado y a otros pares. En esta conexión, algunos mensajes de gestión se pasan vía el servidor y otros mensajes son directamente intercambiados entre los pares.

Los ***sistemas P2P estructurados*** poseen un registro de los datos que pueden ser compartidos así como una función de "*hashing*" que provee la localización de cada par en el mismo espacio de identificación. Tanto los datos como los pares poseen un identificador único (o llave). Con el fin de entregar eficazmente las consultas al par que contiene los datos deseados, cada par maneja una tabla de enrutamiento predefinido o creado a partir de las interacciones previas con otros pares.

En cuanto al tratamiento de las consultas en los *sistemas P2P*, Vdovjak *et al.* [19] describen arquitecturas y *sistemas P2P* que someten consultas a fuentes de datos *RDF*. Este trabajo muestra cómo, en *sistemas P2P* tales como *Gnutella*[1] y *Freenet*[2], los recursos son alcanzados utilizando una estrategia de enrutamiento genérico (pasando de un par vecino a otro par vecino hasta encontrar el recurso). Otros sistemas tales como *P-Grid*[3] y *CAN*[4] utilizan índices para la búsqueda de información. En sistemas como *Edutella*[5] y *Piazza*[6], cada par sólo conoce los pares con los cuales se puede comunicar. Igualmente, sistemas tales como *Napster*[7] y *Gnutella* utilizan la replicación de ítems a través de los pares como un mecanismo de tolerancia a fallas. Por su parte, Kokkinidis *et al.* [9] presentan diferentes proyectos basados en la aproximación *P2P* que asignan planes a cada par para ejecutar tareas tales como el enrutamiento y el tratamiento de consultas a través de la red. Entre los proyectos citados por estos autores, se encuentran: *Mutant Query Plans*[8], *AmbientDB*[9] o aún *Edutella* que explora el diseño y la implementación de un *schema* basado en una infraestructura *P2P* para la Web semántica. En *Edutella*, el contenido de cada par es descrito por *schemas RDF* y los super-pares son responsables del enrutamiento de mensajes y de la integración/mediación de las bases de datos de los pares.

Con el fin de proveer a los usuarios nómadas con la información mejor adaptada a sus características, una aplicación que se ejecuta en *DM* debe utilizar mecanismos para propagar las consultas del usuario hacia las fuentes más apropiadas capaces de responderlas y que toman en cuenta las preferencias del usuario, las características de su *DM*, su localización, *etc.* [13] [15]. Este es el principal objetivo del proceso de *enrutamiento de consultas* que será tratado en la sección siguiente.

III. PROCESO DE ENRUTAMIENTO DE CONSULTAS EN LOS SISTEMAS P2P

Xu *et al.*, [20] definen el *enrutamiento de consultas* como el problema general que se basa en dos actividades principales: de una parte, la evaluación de una consulta utilizando las fuentes más relevantes, y de otra parte, la integración de los resultados provenientes de las fuentes. Con el fin de llevar a cabo estas dos actividades, estos autores identifican tres subproblemas: *i)* la selección de las fuentes de información que necesita analizar la consulta del usuario con el fin de determinar aquellas fuentes que son aptas a responderla. Con el fin de resolver este subproblema, el sistema debe conocer la información manejada por cada fuente; *ii)* el segundo subproblema concierne la evaluación de la consulta del usuario por parte de las fuentes de información escogidas en la etapa de selección. Debido a eventuales

[1] *Gnutella* : http://www.gnutella.com/

[2] *Freenet* : http://freenetproject.org/

[3] *P-Grid* : http://www.p-grid.org/

[4] *CAN*: http://www.can-cia.org/

[5] *Edutella* : http://edutella.jxta.org/

[6] *Piazza* : http://data.cs.washington.edu/p2p/piazza/

[7] *Napster* : http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p4.html

[8] *Mutant Query Plans* : http://web.cecs.pdx.edu/~vpapad/mqp.html

[9] *AmbientDB* : http://homepages.cwi.nl/~boncz/ambientdb.html

representaciones heterogéneas de los datos en las diferentes fuentes, la consulta original deberá ser traducida en el vocabulario de las fuentes interrogadas; *iii)* el tercer subproblema concierne la fusión de los resultados y de hecho, también se refiere a la integración de los resultados provenientes de las diferentes fuentes de información.

Este artículo se concentra en el subproblema de la selección de las fuentes de información hacia las cuales redirigir las consultas. Se presentan algunos de los trabajos que tratan este subproblema y se clasifican en tres categorías: aquellos que utilizan un reagrupamiento de pares con el objetivo responder las consultas (los pares de un mismo grupo son susceptibles de responder consultas similares); aquellos trabajos basados en *filtros* y la *correspondencia de datos* ("*matching*") entre los términos de una consulta y los datos de una fuente de información; finalmente se presentan algunos trabajos basados en estrategias de *Confianza y Reputación* ("*Trust and Reputation*").

A. Estrategias que reagrupan super- pares/pares

Entre los trabajos que reagrupan pares capaces de responder las consultas, aquel de Brunkhorst *et al.* [2] permite especificar los pares asociados a un *super-par* y la información administrada por cada uno de ellos en el marco de los *sistemas P2P* basados en *schemas*. Los *super-pares* componen un subconjunto de pares, cada uno poseedor de responsabilidades específicas y de la capacidad de reagruparse con otros, con el fin de llevar a cabo tareas tales como el enrutamiento de consultas, la mediación en la resolución de conflictos y el trabajo en grupo. Estos autores definen un conjunto de índices necesarios para el enrutamiento de consultas entre los (super-) pares del sistema. Kokkinidis *et al.* [9] proponen reagrupar los pares que comparten los mismos *schemas* de información para construir planes con el fin de responder las consultas de una manera distribuida. En este trabajo, las tareas de un par y aquellas de un super-par están bien definidas. Los *super-pares* son los responsables del enrutamiento de consultas, y los *pares* son responsables del tratamiento des consultas (es decir, de su ejecución).

B. Filtros y mecanismos de correspondencia de datos ("Matching")

Koloniari *et al.* [10] definen el enrutamiento de consultas como un mecanismo de determinación de la localización de los nodos que contienen los documentos que responden las consultas. Esta localización se basa en un mecanismo de *correspondencia* de documentos y de *filtros*. Los *filtros* son estructuras de datos especializadas que reagrupan la información general de grandes colecciones de documentos y que redirigen las consultas solamente hacia los nodos que pueden contener la información pertinente. Cada nodo mantiene dos tipos de filtros: el *filtro local* ("*local filter*") que reagrupa la información general de los documentos almacenados localmente en el nodo, y los *filtros fusionados* ("*merged filtres*") que reagrupan la información general de los documentos de los nodos vecinos. Cuando una consulta llega a un nodo, este nodo verifica su filtro local y utiliza los filtros fusionados para dirigir la consulta únicamente hacia los nodos cuyos filtros hacen correspondencia con las consultas.

Otro sistema que utiliza *filtros* para la gestión de consultas es aquel propuesto por Idreos *et al.* [7] quienes presentan un algoritmo que filtra los documentos necesarios para responder una consulta. Este algoritmo utiliza dos tablas llamadas "*Total*" y "*Count*". Para cada consulta, *Total* almacena el número de fórmulas atómicas contenidas en esta consulta. En cuanto a *Count*, esta tabla es utilizada para contar el número de fórmulas atómicas evaluadas como verdaderas para un documento. Cada elemento de estas tablas es inicializado a cero al comienzo del algoritmo. Si al final del algoritmo, una entrada de la consulta en la tabla *Total* es igual a su entrada en la tabla *Count*, entonces el documento responde la consulta.

Xu *et al.* [20] presentan algunas estrategias basadas en técnicas de "*clusters*" para la selección de las fuentes de información. Estas estrategias se componen de dos etapas: por un lado, la construcción del conocimiento sobre las fuentes que componen cada "*cluster*" y sobre la información almacenada por cada una. Por otro lado, se efectúa una clasificación de las fuentes a través de un puntaje de correspondencia. Dicho puntaje ("*score*") de correspondencia ("*matching*") es calculado por cada *cluster*. Cuando una consulta corresponde a un *cluster*, es normal que varias *fuentes* de este *cluster* respondan de una manera pertinente la consulta. Además, cuando una consulta corresponde a un número significativo de *clusters* a los cuales pertenece una fuente de información, se puede suponer que esta fuente es muy pertinente con respecto a la consulta. Con esto, la posición de cada fuente en la clasificación es determinada teniendo en cuenta la suma de los puntajes de la correspondencia obtenidos por los *clusters* a los cuales la fuente pertenece, y finalmente es ponderada por el tamaño de estos *clusters*.

C. Estrategias de Confianza y Reputación ("Trust and Reputation")

Con el fin de optimizar el proceso de enrutamiento de consultas, Agostini *et al.* [1] proponen utilizar diferentes métricas ligadas a la confiabilidad de las fuentes, a sus capacidades de satisfacer las necesidades de información del usuario y a su confiabilidad para entregar la información a los usuarios. Agostini *et al.* proponen una estrategia de *Confianza y Reputación* permitiendo seleccionar los pares hacia los cuales redirigir las consultas. La *confianza* asignada a un par es el grado de seguridad estimado con respecto a la calidad de sus respuestas. La *confianza* se basa en su *reputación*. La *reputación* de un par es la opinión que los otros pares tienen acerca de él; esta opinión se construye a partir de los resultados provistos por un par para consultas

previas, considerando criterios tales como la rapidez en responder, lo exhaustivo de las respuestas, la pertinencia de las respuestas, entre otros. La *reputación* es un valor sobre una escala cualitativa o cuantitativa calculada por cada agente. Un sistema de reputación de pares recolecta, distribuye y maneja la información sobre el comportamiento pasado de los pares. La estrategia de *Confianza y Reputación* propuesta por Agostini *et al.* se basa en el proceso siguiente: un componente ***buscador*** (*"seeker"*) selecciona los pares capaces de responder la consulta *R* con la más alta probabilidad considerando criterios establecidos (por ejemplo, la rapidez en responder, la cantidad de información provista/entregada, *etc.*). Con el fin de decidir, el *buscador* construye y almacena una lista $<p_1, p_2,\ldots p_k>$ de pares dignos de confianza a quienes someter la consulta. La lista está ordenada utilizando un nivel de confianza decreciente. La estrategia seguida por el *buscador* con el fin de responder una consulta es la siguiente: primero que todo, solicita a p_1, después a p_2, y continúa en tanto que recibe respuestas pertinentes. Vale la pena notar, que la lista de pares dignos de confianza puede evolucionar.

Una estrategia similar es aquella presentada por Yang *et al.* [21] quienes introducen el concepto de "*Guía de Enrutamiento*" ("*Routing Guide*") que se basa en los resultados obtenidos de las consultas previamente tratadas; dicha guía es utilizada con el fin de determinar los caminos de enrutamiento para las siguientes consultas.

Röhm *et al.* [14] proponen un coordinador simple e inteligente que puede llevar a cabo diferentes tareas. El coordinador puede también dirigir las consultas hacia un componente de su elección, descomponer y dirigir consultas complejas a diferentes componentes en el caso de la partición o de la replicación, y dirigir e implementar la replicación. Röhm *et al.* utilizan el *enrutamiento de consultas basado en el equilibrio* del número de consultas que consiste en dirigir una consulta hacia el componente que tiene el mínimo de consultas activas, o el *enrutamiento de consultas basado en la afinidad* que consiste tanto en agrupar las consultas accediendo los mismos datos como en dirigir este grupo hacia los componentes que manejan estos datos. La *afinidad* entre dos consultas es determinada por un análisis tanto de los datos consultados así como de las consultas de las sesiones previas.

En un ambiente nómada, la información solicitada por un usuario puede evolucionar en función de las características del *contexto de uso*, tales como la localización y el momento de conexión. Por ejemplo, un usuario puede solicitar a un sistema, una lista de restaurantes y desear implícitamente obtener aquellos localizados en la calle donde él se encuentra y que están abiertos en el momento en el cual se emite esta consulta. Las técnicas de enrutamiento de consultas presentadas anteriormente no permiten, por sí solas, manejar este dinamismo del *contexto de uso* en el cual el usuario interroga el sistema. La siguiente sección presenta *PUMAS*, nuestro framework que toma en cuenta estos aspectos, particularmente gracias a mecanismos de enriquecimiento de la consulta inicial.

IV. El framework PUMAS

Durante un proceso de búsqueda de información en ambientes nómades, un usuario puede afrontar diferentes problemas tales como: dificultades de acceso a las fuentes de información, ligadas a las características de red y de su dispositivo de acceso; la falta de adaptación de la información al usuario con respecto tanto a sus características y a sus preferencias, así como a las restricciones técnicas de su dispositivo de acceso; la falta de mecanismos para la búsqueda de información distribuida en diferentes fuentes de información que se encuentran almacenadas en diversos tipos de dispositivos (servidores o dispositivos móviles (*DM*)). Con el fin de resolver estos problemas, se ha propuesto *PUMAS*, un framework basado en agentes que provee al usuario nómada, la información adaptada a sus características y a aquellas de su dispositivo de acceso, sin importar el tipo (móvil o no). *PUMAS* ofrece igualmente los medios de interrogar diversas fuentes, concernientes a *Sistemas de Información* (*SI*) localizados en servidores, o simples archivos almacenados en un dispositivo móvil de tipo *PDA* por ejemplo. Lo que resta de este artículo se concentra en la consulta de las fuentes de tipo "*Sistema de Información*" (*SI*).

La arquitectura de *PUMAS* se compone de cuatro *SMA* (ver Figura 1) que se describen brevemente. Su descripción completa puede encontrarse en [3]:

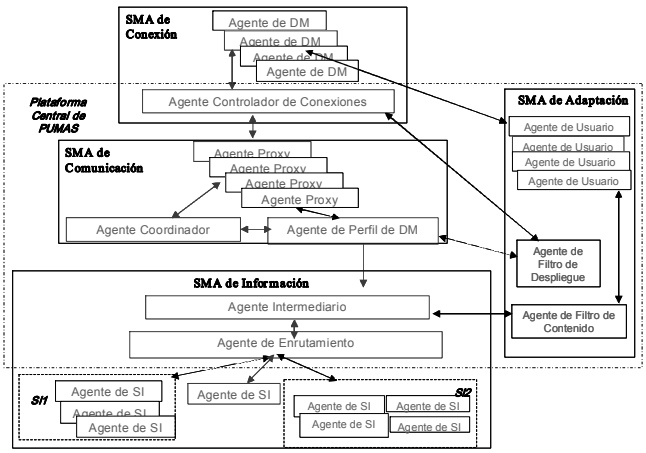

Figura 1. Arquitectura lógica de PUMAS.

- El *SMA de conexión* provee mecanismos para facilitar la conexión de diferentes tipos de *DM* a los *SI*.
- El *SMA de comunicación* asegura una comunicación transparente entre el *DM* y el sistema, y aplica un filtro de despliegue (con la ayuda de los agentes del *SMA de adaptación*) con el fin de presentar la información de una manera adaptada tomando en cuenta las restricciones técnicas del *DM* del usuario.
- El *SMA de información* recibe las consultas de los usuarios, las redirige a los *SI* capaces de responderlas,

aplica un filtro de contenido (con la ayuda de los agentes del *SMA de adaptación*) teniendo en cuenta el perfil del usuario en el sistema y retorna los resultados al *SMA de comunicación*.

– El *SMA de adaptación* que se comunica con agentes de los otros tres *SMA* con el fin de intercambiar información sobre el usuario, las características de conexión y de comunicación, las características del *DM*, *etc*.

Las características principales de los *sistemas P2P híbridos* que se pueden identificar en *PUMAS* son [3]: *i)* Un *DM* puede comunicarse con un *SI* específico (situado en un servidor o en un *DM*) pasando el identificador de este sistema como un parámetro de la consulta. El *agente de enrutamiento* transmite entonces la consulta a este *SI* específico (lo que se considera como una comunicación de agente a agente) y *ii)* los agentes tienen la autonomía de conectarse a *PUMAS* y de desconectarse.

V. Tratamiento de consultas en PUMAS

Esta sección presenta tres escenarios asociados al tratamiento de una consulta en *PUMAS*. En nuestra propuesta, las interacciones entre los agentes se basan en los mensajes intercambiados y corresponden a actos de comunicación [11], por ejemplo, "*propose*", "*request*", "*subscribe*", "*propagate*", *etc.* Los archivos mencionados en este artículo están escritos en *OWL,* siguiendo las extensiones de *CC/PP* propuestas por Indulska *et al.*[8] para involucrar en estos archivos características del usuario, de su sesión, de su *DM* y de su localización. Se presenta igualmente el enrutamiento de consultas tal como se lleva a cabo en *PUMAS* una vez el *agente de enrutamiento* (perteneciente al *SMA de información*) recibe una consulta (ver Figura 1). Ejemplos tomados del medio hospitalario son propuestos para ilustrar nuestra propuesta.

A. Escenario asociado a la conexión

Cuando un usuario se conecta la primera vez al sistema utilizando su *DM,* el *agente de DM*, que se ejecuta en el *DM* del usuario, envía un mensaje "*propose*" (proposición de conexión) al *agente controlador de conexiones*. Si no hay un *agente proxy* que represente este *agente de DM*, el *agente controlador de conexiones* crea un *agente proxy* y envía un mensaje "*subscribe*" al *agente coordinador* con el objetivo de que el *agente proxy* sea inscrito en el sistema. El *agente coordinador* informa al *agente de perfil de DM* de esta inscripción. El *agente controlador de conexión* envía igualmente al *agente de DM* un mensaje "*confirms*" cuando el proceso de inscripción ha finalizado (correctamente o no) (ver Figura 2).

Cuando el *agente de DM* recibe el mensaje de confirmación, un *agente de usuario* es creado en la plataforma central de *PUMAS* con el fin de manejar el perfil del usuario. Este perfil es definido en el archivo de *sesión* actual, administrado por el *agente de DM* y enviado al *agente de usuario*. El *agente de DM* envía también al *agente de usuario* un archivo de *usuario* que contiene las *preferencias del usuario* para esta sesión. El *agente de DM* envía dos archivos al *agente controlador de conexiones* con el fin de hacer conocer las características del *DM* y aquellas de su localización (los archivos respectivamente de *dispositivo* y de *localización*).

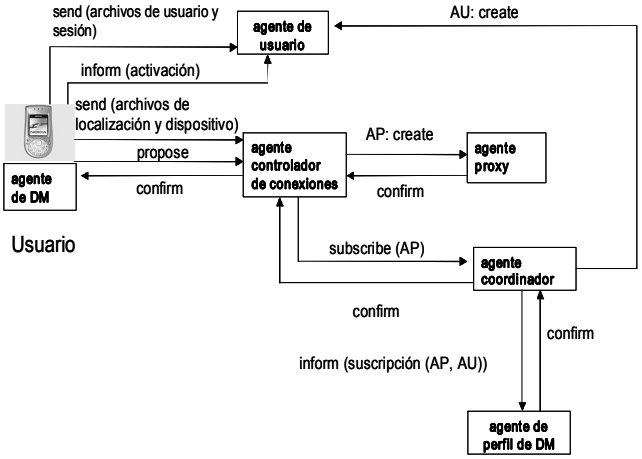

Figura 2. Escenario asociado a la conexión.

El *agente de DM* puede ser obligado a enviar nuevamente estos archivos y de esta manera notificar los cambios ocurridos. Así mismo, cuando el usuario cambia de localización, de dispositivo o desea expresar preferencias específicas para la sesión actual, el *agente de DM* envía respectivamente los archivos de *localización*, de *dispositivo* o de *usuario*. Este reenvío de archivos es particularmente utilizado durante el tratamiento de una consulta con el fin de adaptar la información utilizando nuestra propuesta.

En la sección siguiente, se muestra el escenario correspondiente al envío de una consulta.

B. Escenario de envío de una consulta

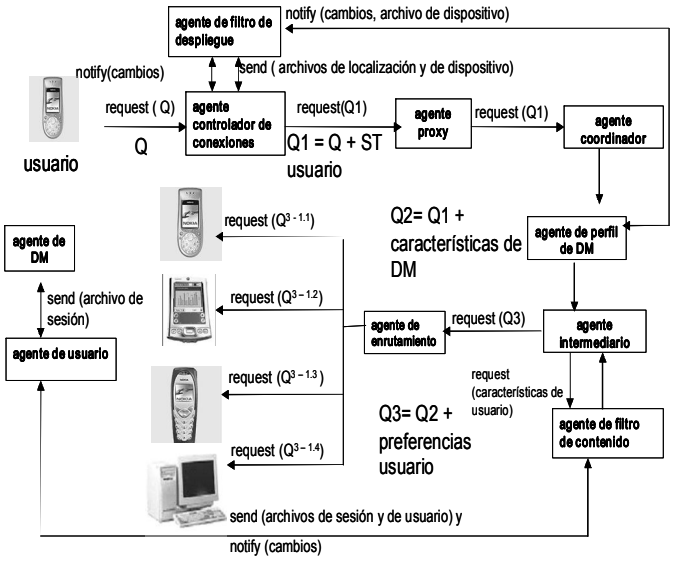

Figura 3. Escenario de envío de una consulta.

Cuando un usuario envía una consulta de información *Q* (ver Figura 3), el *agente de DM* la transmite al *agente controlador de conexiones*. Si este agente ha recibido nuevos archivos de localización y/o de dispositivo, los envía al *agente de filtro de despliegue*. Si el usuario ha establecido como preferencia (definida en el archivo de *usuario*) el hecho de que su consulta *Q* depende simultáneamente de su localización y del tiempo de conexión, el *agente controlador de conexiones* adiciona a *Q* la información sobre el momento de la conexión, la localización del usuario y las características de conexión del *DM* del usuario. Esto conduce a la creación de una nueva consulta *Q1* (ver Figura 3), definida por *Q1 = Q + características Espacio-Temporales* (*ST*) del usuario. Nuestra propuesta se ilustra mediante un ejemplo que pone en escena un médico quien desea consultar su agenda (que corresponde a una consulta *Q*) y que ha definido esta actividad como dependiente de la localización y del momento de la conexión. Según su localización (puede encontrarse en el hospital o en su consultorio), el médico desea consultar las citas médicas que él tiene para ese lugar y puede suponerse, que sólo le interesan las citas que tiene de aquí a dos horas. *PUMAS* le permite consultar esta información sin necesidad de que él precise ni el lugar ni el momento de la conexión. La localización efectiva del médico así como el criterio temporal (fecha del sistema + 2 horas) son adicionados a *Q* para crear la consulta enriquecida *Q1*. La consulta *Q1* es enseguida enviada al *agente proxy*. *Q1* es transmitida al *agente coordinador*, después al *agente de perfil de DM*. Este agente recibe también del *agente de filtro de despliegue*, el archivo de *dispositivo*, si este último ha recibido un nuevo archivo de *dispositivo* o notificaciones sobre cambios en el dispositivo. El *agente de perfil de DM* adiciona a *Q1* características ligadas al *DM*. Estas características son provistas por el *agente de filtro de despliegue* quien las ha deducido de consultas previas o las ha extraído de su base de conocimiento. Por ejemplo, si el médico está conectado a través de una *Palm Tungsten C*, que no soporta imágenes, las respuestas a las consultas del médico no podrán desplegarse sino a manera de texto. El *agente de perfil de DM* solicita al *agente de filtro de despliegue* la información sobre este *DM*. El *agente de perfil de DM* podría recibir del *agente de filtro de despliegue* hechos definidos de la siguiente manera (de acuerdo a la definición presentada en [4], ver Figura 4):

```
(deffacts CaracterísticaDM
(TipoDM "Palm Tungsten C")
(característica
        (tipo "video_no_soportado")
        (descripción "red_Wi-Fi"))
(característica
        (tipo "varias_imágenes")
        (descripción "red_Wi-Fi")
(característica
        (tipo "texto")
        ( descripción "red_Wi-Fi" "Bluetooth")))
```

Figura 4. Hecho que define una característica de DM.

La nueva consulta *Q2* (ver Figura 3), definida por *Q2 = Q1 + características de DM*, es enviada por el *agente de perfil de DM* a un *agente intermediario*. El *agente intermediario* adiciona a *Q2* las características específicas del usuario en el sistema. Estas características son solicitadas al *agente de filtro de contenido* (ver Figura 3). *Q3* es definida por *Q3 = Q2 + preferencias usuario*. Por ejemplo, un médico puede expresar sus preferencias de información a través de una interfaz. Estas preferencias son traducidas por el *agente de usuario* como se describe aquí abajo.

Un ejemplo de preferencia de información es la siguiente[10] : "*Cada vez que solicite los exámenes clínicos de un paciente, el sistema deberá también proveerme su dieta y los medicamentos que le han sido prescritos; prefiero los resultados en un formato gráfico; pero si mi DM no soporta este formato, me gustaría recibirlos como un texto*".

Esta preferencia puede ser traducida por el siguiente hecho[11] (almacenado en la base de conocimiento del *agente de usuario,* ver Figura **5**):

```
(1)  (deffacts Preferencia_Información_Aumentada
(2)  (IDUsuario "Doctor Jaime Pérez")
(3)  (info_requerida "exámenes clínicos")
(4)  (info_complementaria "dieta" "medicamentos prescritos")
(5)  (acción
(6)  (nombre "mostrar")
(7)  (atributo (nombre "orden")
(8)  (ítems "exámenes clínicos" "dieta" "medicamentos
prescritos"))
(9)  (atributo (nombre "formato_gráfico")
(10) (ítems "JPEG"))
(11) (problema
(12) (nombre "Multimedia no Suportada por DM")
(13) (tipo "incompatibilidad")
(14) (causas "Solamente Archivos Texto Suportados"))
(15) (acción
(16) (nombre "mostrar")
(17) (atributo (nombre "orden")
(18) (ítems "exámenes clínicos" "medicamentos prescritos"
"dieta"))
(19) (atributo (nombre "formato_texto") (ítems "XML" "txt")))
```

Figura 5. Hecho que define una preferencia de información aumentada.

Las líneas (1) a (4) corresponden al hecho de una *preferencia de información* simple (la *info_requerida* corresponde a los exámenes clínicos y la *info_complementaria* corresponde a la dieta y a los medicamentos prescritos a un paciente). Las líneas (5) a (10) conciernen la manera (acción) en la que el sistema debe actuar para desplegar la información en el dispositivo de acceso del usuario. El sistema debe desplegar la información en un orden específico utilizando un formato gráfico. De la

[10] Un tratamiento de preferencias más detallado se presenta en [4] y en [6].

[11] Un hecho es una pieza de conocimiento que describe un elemento del mundo real. En este trabajo, los hechos son descritos en *JESS*. Con el fin de clarificar su definición, mostramos igualmente su representación como clases *UML*.

línea (11) a la línea (14), se ilustra cómo puede ser descrito un problema particular (que puede ocurrir) y las líneas (15) a (19) conciernen la manera (acción) en la cual el sistema debe reaccionar si este problema ocurre. La acción corresponde a la manera en la cual el sistema debe desplegar la información, aquí en un orden específico utilizando formato texto.

El *agente de DM* envía al *agente de usuario* un nuevo archivo de *usuario* (que contiene sus preferencias) en el caso en el cual el usuario exprese nuevas preferencias para la sesión actual. El *agente de usuario* comunica las nuevas preferencias al *agente de filtro de contenido* con el fin de generar un perfil único para este usuario, dando prioridad a las nuevas preferencias para esta sesión. El *agente intermediario* transmite *Q3* al *agente de enrutamiento* encargado de llevar a cabo el proceso de *enrutamiento de consultas* que se describe en la siguiente sección.

C. Proceso de enrutamiento de consultas en PUMAS

En esta sección, se explican e ilustran las actividades (inspiradas en Xu *et al.* [20]) ligadas al enrutamiento de consultas en *PUMAS*. Este proceso es ejecutado por los *agentes de enrutamiento* (pertenecientes al *SMA de información*) quienes reciben las consultas enriquecidas como se describió anteriormente. Cuando se recibe une consulta, un *agente de enrutamiento* puede enviarla a un *agente de SI* específico, duplicarla y enviarla a varios *agentes de SI*, o el *agente de enrutamiento* puede descomponer la consulta en subconsultas enviadas a uno o varios *agentes de SI*. La Figura 3 muestra por ejemplo, un escenario en el cual *Q3* es descompuesta en $Q^{3-1.1}$, $Q^{3-1.2}$, $Q^{3-1.3}$ y $Q^{3-1.4}$ y son enviadas a los *agentes de SI* ejecutándose en un servidor y en diferentes *DM*. Lo que resta del artículo se concentra en el caso en el que se debe descomponer la consulta. Otros ejemplos tales como la transmisión de una consulta de punto a punto (*i.e.* de *agente de DM* a *agente de DM*) se presentan en [3].

1) Análisis de la consulta: El *agente de enrutamiento* analiza la complejidad de la consulta. Una consulta es considerada como "*simple*" si puede ser tratada por solamente un *agente de SI* y "*compleja"* si varios *agentes de SI* son requeridos. En este caso, esta actividad dará lugar a la descomposición de la consulta en subconsultas. Este análisis es más precisamente basado en los hechos relativos a los *SI*, almacenados en una base de conocimiento manejada por el *agente de enrutamiento*. Este agente analiza igualmente los criterios de adaptación de una consulta (por ejemplo, la localización, las actividades del usuario), la lista de los destinatarios de una consulta, *etc.* Después de este análisis, el *agente de enrutamiento* decide si debe o no descomponer la consulta en subconsultas. Suponga a manera de ejemplo que un médico desea consultar los resultados de los exámenes clínicos de un paciente, su dieta alimenticia y los medicamentos prescritos, y que ningún *SI* puede responder completamente esta consulta. Esta consulta es considerada como una consulta "*compleja"* la cual es descompuesta por el *agente de enrutamiento* en tres subconsultas que corresponden respectivamente a los "*exámenes clínicos*", la "*dieta*" y los "*medicamentos prescritos*". Tal descomposición se basa en el conocimiento almacenado por el *agente de enrutamiento* con respecto a la información manejada por los diferentes *SI*. Un ejemplo de este conocimiento descrito en *JESS* es dado a continuación y corresponde al *SI* de la farmacia del hospital (ver Figura 6):

```
(assert (SI
(nombre SIFarmacia)
(agenteID ASIFarmacia)
(dispositivo servidor)
(localización " Hospital Norte ")
(items_información
"medicamentos prescritos a pacientes" "dosis" " medicamentos")))
```

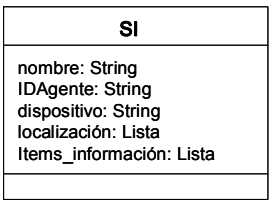

Figura 6. Hecho que define un sistema de información.

Con el fin de descomponer la consulta, el *agente de enrutamiento* identifica los ítems de la consulta [$item_1$, $item_2$... $item_s$] durante la actividad de *análisis de la consulta*. En nuestro ejemplo, la consulta corresponde a: "*los resultados de los exámenes clínicos de un paciente, su dieta alimenticia y los medicamentos prescritos*", y el $item_1$ corresponde a "*exámenes clínicos*", el $item_2$ a "*dieta*" y el $item_3$ a "*medicamentos prescritos*". Enseguida, este agente busca por cada ítem, los *SI* que manejan un item_información equivalente. Dos ítems son "*equivalentes*" si son iguales semánticamente (es decir, si los dos ítems tienen el mismo significado) o sintácticamente (por ejemplo, igualdad de cadenas de caracteres). La relación de equivalencia es definida por los diseñadores y desarrolladores de aplicaciones quienes implementan el algoritmo de correspondencia que explicamos a continuación. El algoritmo de correspondencia (ver Figura 7.) genera una lista *LSI* que contiene tuplas:

(*SI, <lista de ítems de la consulta manejados por el SI>*)

(1) Inicializar *LSI* respondiendo particularmente la consulta (LSI) // *LSI* se inicializa vacía
(2) nSI← 0; // contador de los *SI* manejando ítems equivalentes a aquellos de la consulta
(3) i← 1; // índice sobre los ítems de la consulta
(4) **Mientras que** i ∈ [1, n] **haga //** índice de los ítems de la consulta
(5) j← 1; // índice sobre los *SI*
(6) **Mientras que** j ∈ [1, s] **haga //** índice de los *SI* conocidos por el *agente de enrutamiento*
(7) m← 0; // índice sobre los ítems de información del SI_j
(8) **Mientras que** m < tamaño (lista de ítems de información manejados por SI_j) **haga**
(9) **Si** Comparar ($ítem_s$, $item\_información_m$ manejada por SI_j) **entonces**
(10) nSI ← nSI +1;

(11) LSI← LSI ⊕ (SI_j, <$item_i$>); // adicionar la tupla a LSI
(12) **Fin si**
(13) m ← m + 1;
(14) **Fin Mientras que**
(15) j ← j + 1;
(16) **Fin Mientras que**
(17) i ← i + 1;
(18) **Fin Mientras que**
(19) **Si** nSI es igual 0 **entonces**
(20) tipo_consulta ← *"sin respuesta"*
(21) **sino**
(22) sid← contarDiferentes (LSI); // cuenta el número de *SI* diferentes pertenecientes a *LSI*
(23) **Si** sid es igual a 1 **entonces**
(24) tipo_consulta ← *"simple"*
(25) **sino** // sid es superior a 1
(26) tipo_consulta ← *"compleja"*
(27) **Fin Si**
(28) **Fin Si**
(29) LSI ← Compactar (LSI)

Figura 7. Algoritmo de correspondencia.

Primero que todo, la lista *LSI* está vacía (ver línea (1)). Enseguida, la variable ***nSI*** (que contiene el número de *SI* que manejan los ítems equivalentes a aquellos de la consulta, ver línea (2)) es inicializada. Para cada ítem de la consulta ($ítem_s$ con i ∈ [1, n], ver líneas (3) a (18)), se buscan los SI_j (SI_j con *j* ∈ [1, s]) que manejan ítems de información equivalentes a aquel que está siendo analizado. El método "*Comparar*" prueba si los ítems son equivalentes (ver línea (9)). Si son equivalentes, el algoritmo aumenta la variable ***nSI***, y adiciona a ***LSI*** una tupla cuyo primer término corresponde a SI_j y el segundo corresponde al ítem analizado (ver líneas (10) y (11)). Cuando todos los ítems de la consulta han sido analizados, el algoritmo verifica el valor de ***nSI*** (ver líneas (19) a (27)). Si este valor es cero, esto significa que ningún *SI* maneja ítems equivalentes a aquellos de la consulta. En el caso contrario, el algoritmo analiza la lista *LSI* con el fin de conocer el número de *SI* diferentes que manejan ítems equivalentes a aquellos de la consulta, con el objetivo de atribuir el tipo de la consulta ("*simple*" o "*compleja"*). Este número es calculado utilizando el método "*contarDiferentes*" (ver línea (22)). Finalmente, el algoritmo utiliza el método "*Compactar*" que deja en *LSI* una sola tupla por *SI*. El segundo término de esta tupla corresponde a todos los ítems de la consulta manejados por *SI*.

Con el fin de clarificar el funcionamiento de los métodos "*contarDiferentes*" y "*Compactar*", los ilustramos a través de un ejemplo:

Suponga que el *agente de enrutamiento* ha identificado los ítems: i_1, i_2, i_3, i_4 y i_5, y que él conoce los *SI* siguientes: *SI_1, SI_2, SI_3, SI_4, SI_5 y SI_6*. Después de la comparación de los ítems (ver líneas (4) a (19)), se supone que *LSI* está compuesta de las tuplas siguientes:

(SI_1, <i_1>) (SI_3, <i_1>) (SI_5, <i_1>) (SI_1, <i_2>) (SI_3, <i_2>)
(SI_4, <i_2>) (SI_5, <i_2>) (SI_1, <i_3>) (SI_3, <i_3>) (SI_4, <i_4>)
(SI_5, <i_4>) (SI_1, <i_5>) (SI_3, <i_5>) (SI_4, <i_5>) (SI_5, <i_5>)

El resultado del método "*contarDiferentes*" es 4 ya que en las tuplas sólo aparecen *SI_1, SI_3, SI_4* y *SI_5* (*SI_2* y *SI_6* no aparecen). El método "*Compactar*" deja una sola tupla por *SI*. Después de la ejecución de este método, la *LSI* se compone de las tuplas siguientes:

(SI_1, <i_1, i_2, i_3, i_5>) (SI_3, <i_1, i_2, i_3, i_5 >)
(SI_4, <i_2, i_4, i_5 >) (SI_5, <i_1, i_2, i_4, i_5>)

En el ejemplo, la *LSI* se compone de:

(*SIFarmacia*, < *" medicamentos prescritos"*>)
(*SINutricionista*, < *"dieta"*>)
(*SILaboratorioClínico*, < *"exámenes clínicos"*>)

Después de la ejecución del algoritmo, se puede concluir que la consulta es "***compleja***".

Un ítem de la consulta puede ser manejado por varios *SI*, es entonces necesario seleccionar los *SI* más apropiados para responderlo. La siguiente sección describe esta selección.

2) Selección de Sistemas de información: Una consulta puede ser redirigida hacia un agente específico o hacia un grupo de agentes. Si los destinatarios de una consulta son conocidos, la selección es relativamente fácil. De otra manera, el *agente de enrutamiento* selecciona los *SI* y compone la *red de vecinos* (tomando en cuenta la *LSI*, producto de la actividad previa). Esta selección se basa en las ideas de Yang *et al*. [21]. Estos autores proponen un proceso de recuperación de la información en *sistemas P2P no estructurados*, donde cada nodo posee una colección de datos compartidos con otros nodos. Cuando un usuario envía una consulta, su nodo se convierte en el emisor de la consulta y puede enviar mensajes (que incluyen la consulta) a varios de sus vecinos. Cuando un vecino recibe el mensaje con la consulta, él la trata utilizando, primero que todo, su información local. Si el nodo encuentra algunos resultados, los retorna al nodo emisor. Para *PUMAS*, un par es "*vecino*" de otros pares, si satisface un conjunto de características (criterios definidos en las preferencias del usuario de una aplicación), tales como una localización próxima, actividades semejantes, un rol similar, un conocimiento similar, colegas pertenecientes al mismo grupo, etc. Las características no son, sin embargo, restringidas a criterios de proximidad física.

Se pueden distinguir tres casos para constituir una ***red de vecinos*** en la cual cada nodo es un *SI* pudiendo potencialmente responder una consulta.

El caso más simple es aquel donde una consulta puede ser únicamente tratada por un agente. El *agente de enrutamiento* utiliza su base de conocimiento con el fin de contactar el *SI* que puede responder la consulta.

En el segundo caso, varios agentes pueden responder la misma consulta. La manera más simple de conformar esta red es agrupar todos estos agentes. Este agrupamiento es útil

cuando el *agente de enrutamiento* no posee ninguna información sobre los *SI* o cuando es la primera vez que este agente trabaja con los vecinos. Con el fin de evitar comunicaciones inútiles, redundantes o inutilizables, y de seleccionar los vecinos más pertinentes, el *agente de enrutamiento* aplica criterios de adaptación de la consulta. Por ejemplo, si el criterio es la localización, la red se compone de los vecinos más próximos; si las consultas del usuario dependen de sus consultas previas, el *agente de enrutamiento* debe redirigirlas a los vecinos más dignos de confianza ("*trusted*"); si el criterio es la similitud, la red debe estar compuesta de los vecinos con un perfil similar, que ejecutan tareas similares, *etc*. Si no hay criterios establecidos, el *agente de enrutamiento* analiza el nivel de confianza ("*trust*") de sus vecinos. El *agente de enrutamiento* asocia un nivel de confianza a cada vecino a partir de las respuestas previas. Para asignar dicho nivel se sigue la estrategia de *Confianza y Reputación* ("*Trust and Reputation*") propuesta por Agostini *et al.* [1] (ver sección *III.C*).

En el tercer caso, una consulta ha sido descompuesta en varias subconsultas durante la etapa de análisis. El *agente de enrutamiento* analiza qué agentes pueden responder a cada subconsulta. Dichos agentes constituyen la *red de vecinos*. Para cada subconsulta, se efectúa la selección del o de los *SI* aptos para responder (es decir, se retoma el primer o segundo caso para cada subconsulta). Finalmente, la *red de vecinos* está compuesta de la agregación de las diferentes subredes de vecinos, generadas para cada subconsulta.

Para el ejemplo que se ha tratado a lo largo de la sección, el *agente de enrutamiento* toma en cuenta la *LSI* producida en la etapa de análisis. Este agente selecciona como *SI* el de la *Farmacia*, el del *Laboratorio Clínico* y el de los *Nutricionistas* del hospital, los cuales son los únicos capaces de responder a cada una de las tres subconsultas identificadas y generadas. Estas subconsultas son redirigidas a los *agentes de SI* de los tres *SI*. A continuación se presenta el ejemplo de la subconsulta *Consulta*$_1$ producida para el *SI* del Laboratorio Clínico cuyo *agente de SI* se llama *ASILaboratorioClínico*. Según un principio equivalente, las Consulta$_2$ y Consulta$_3$ serán redireccionadas respectivamente a los agentes *ASINutricionista* del *SI* Nutricionista y *ASIFarmacia* del *SI* Farmacia (ver Figura 8).

(*assert* (*Consulta*
(*IDConsulta ConsultaI*)
(*IDUsuario "Doctor Jaime Perez"*)
(*ASI "ASILaboratorioClínico"*)
(*SI "SILaboratorioClínico"*)
(*Info_requerida "Exámenes Clínicos"*)
(*Parámetros "nombre paciente" "fecha"*)))

Consulta
IDConsulta: String IDUsuario: String ASI: String Info_requerida: Lista Parámetros: Lista

Figura 8. Hecho que define una consulta.

3) Redirección de la consulta: Una vez el *agente de enrutamiento* ha identificado los *SI* (vecinos) potenciales, debe analizar el nivel de confianza asociado a cada uno de ellos para determinar un protocolo de redirección de la consulta. Esta información sobre el nivel de confianza puede no estar disponible, si es la primera vez que el *agente de enrutamiento* ejecuta esta consulta, o que él trabaja con estos *SI*. De la misma manera, puede presentarse que los niveles de confianza de los vecinos sean iguales. En estas condiciones (*i.e.*, ausencia de nivel de confianza establecida o de confianza no discriminante), el *agente de enrutamiento* envía la consulta en "*broadcast*", es decir, a todos los vecinos.

Cuando se obtiene una información acerca de la confianza de los *agentes de SI*, el *agente de enrutamiento* redirige la consulta de manera secuencial, comenzando por el agente más digno de confianza. La respuesta a la consulta será aquella del primer agente que responda. Si el agente de enrutamiento no recibe ninguna respuesta, el usuario será informado del fracaso para responder la consulta.

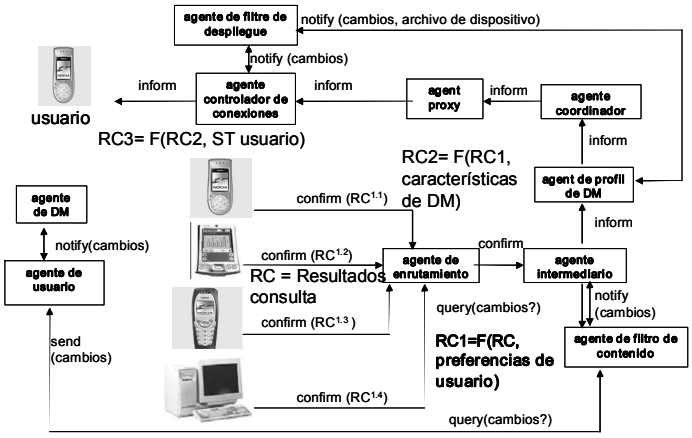

Figura 9. Escenario de recepción de los resultados de una consulta.

Si el *agente de enrutamiento* conoce los únicos vecinos capaces de responder a cada una de las subconsultas (consulta$_1$, …, consulta$_N$), el *agente de enrutamiento* se las envía directamente (y respectivamente). En nuestro ejemplo, el *agente de enrutamiento* envía la Consulta$_1$ al *ASILaboratorioClínico*, la Consulta$_2$ al *ASINutricionista* y la Consulta$_3$ al *ASIFarmacia*.

El *agente de enrutamiento* debe enseguida recolectar las respuestas obtenidas de los diferentes *agentes de SI* y seleccionar las más pertinentes teniendo en cuenta los criterios de adaptación establecidos para la consulta. La siguiente sección presenta la ruta seguida por los resultados de las consultas (desde los *agentes de SI* hasta el *agente de DM*).

D. Escenario de recepción de los resultados de una consulta

Cuando el *agente de enrutamiento* recibe todos los resultados de la consulta de los *agentes de SI* (en el ejemplo de la Figura 9, el *agente de enrutamiento* recibe los resultados parciales – $RC^{1.1}$, $RC^{1.2}$, $RC^{1.3}$ y $RC^{1.4}$ – enviados por los

agentes de SI ejecutándose en un servidor y en diferentes *DM*), los analiza antes de enviar un mensaje de "*confirms*" o de "*disconfirms*" o de "*not understand*" al *agente intermediario*. Este mensaje incluye los resultados de la consulta (*RC*). El *agente de filtro de contenido* envía el archivo de *usuario* al *agente intermediario* en el momento en que se produzcan cambios en las preferencias del usuario. El *agente intermediario* verifica que los resultados pueden satisfacer las *preferencias del usuario* (en el ejemplo de la Figura 9, *RC1* es el resultado de la aplicación del *filtro de contenido* a *RC* teniendo en cuenta las preferencias y el histórico del usuario). El *agente intermediario* redirige *RC1* al *agente de perfil de DM*. El *agente de filtro de despliegue* envía el archivo de *dispositivo* al *agente de perfil de DM* cuando se produzcan cambios en las características del *DM*. Este agente verifica si los resultados pueden ser desplegados teniendo en cuenta las características del *DM* y procede a la primera etapa del *filtro de despliegue* (en el ejemplo de la Figura 9, *RC2* es producido por el filtro de *RC1* considerando las características del *DM*). Enseguida, *RC2* es transmitido por el *agente coordinador* al *agente proxy* y luego al *agente controlador de conexiones*. El *agente de DM* envía el archivo de *localización* al *agente controlador de conexiones* si se producen cambios en la localización del usuario o en las características de conexión del *DM*. El *agente controlador de conexiones* procede a la última etapa del *filtro de despliegue* teniendo en cuenta las características de conexión del *DM* del usuario (*e.g.*, si el usuario está aún conectado, si ha cambiado de localización, si un "*timeout*" se ha cumplido, *etc.*). Gracias a los filtros de *contenido* y de *despliegue*, los resultados de la consulta recibidos por el *agente de DM* son desplegados en el *DM* del usuario. Estos resultados corresponden a la información mejor adaptada al *contexto de uso*, a las *preferencias del usuario*, a las características del *DM* y a la consulta.

Es importante notar que el escenario descrito aquí arriba incluye diferentes etapas de verificación de resultados que pueden parecer inútiles puesto que el escenario de envío de una consulta (ver sección *V.B*) ha permitido refinar la consulta teniendo en cuenta características del usuario y aquellas de su *DM*. Sin embargo, esta información suplementaria, adicionada durante el escenario de envío de una consulta, podría no ser más válida en el momento de entrega de los resultados. Con esto, vale la pena resaltar que las características del usuario (cambios de localización y de preferencias) y los medios de conexión y de comunicación (variación del ancho de banda, diferentes *DM* utilizados, problemas de red) podrían haber evolucionado y en consecuencia, podrían tener un impacto en los resultados a ser entregados al usuario. Los controles presentados en el escenario "*recepción de los resultados de una consulta*" se orientan a eliminar la transmisión de información no pertinente al usuario. También vale la pena notar, que las políticas de tratamiento de las consultas deben ser descritas e integradas al sistema para evitar, por ejemplo, que sean tomados en cuenta ciertos cambios (*e.g.*, un cambio de dispositivo o de localización) que el usuario no desea que sean tenidos en cuenta.

VI. Conclusiones y Trabajo Futuro

Cuando un usuario formula consultas, los resultados pueden provenir de diferentes *Fuentes de Información*. En este artículo, se define un proceso de *enrutamiento de consultas* como un mecanismo que analiza la consulta identificando sus ítems, y haciendo la correspondencia (semántica o sintáctica) entre los ítems y la información administrada por los *SI* conocidos por el *agente de enrutamiento* de *PUMAS*, lo anterior con el fin de seleccionar el(los) *SI* capaz(ces) de responderlas. Después de la identificación de estos ítems y del reconocimiento de los *SI* que manejan ítems equivalentes, este proceso descompone la consulta. Un *agente de enrutamiento* tiene en cuenta criterios de adaptación provistos por el usuario (tales como la localización, las actividades ejecutadas por el usuario durante un periodo, sus preferencias, *etc.*) con el fin de escoger los *SI* más apropiados para responder la consulta. Finalmente, este proceso debe recolectar y agrupar los resultados de la consulta. En este artículo se han igualmente descrito tres escenarios correspondientes a la conexión de un usuario a *PUMAS*, al envío de una consulta, y a la recepción de los resultados. De igual manera se ilustra el proceso de enrutamiento de consultas así como los escenarios a través de un ejemplo que involucra diferentes *SI* de un hospital en el cual un médico solicita información sobre las prescripciones, los exámenes clínicos y la dieta alimenticia de un paciente.

Nuestro trabajo futuro se orienta a la definición del algoritmo para la recolección y el análisis de los resultados provenientes de una o varias *fuentes de información*. Esta es entonces la etapa que sigue al proceso de enrutamiento de consultas aquí descrito.

Agradecimientos

La autora Angela Carrillo Ramos agradece a Fernando De la Rosa, profesor asociado de la Universidad de los Andes (Bogotá, Colombia), por los comentarios y correcciones efectuados a este artículo. Este artículo es el resultado de una parte de la tesis doctoral de la autora en mención, desarrollada en el equipo *STEAMER* del *Laboratorio de Informática de Grenoble* (Francia). El título de doctorado fue obtenido en la Universidad Joseph Fourier en Grenoble (Francia).

Referencias

[1] Agostini, A., Moro, G. (2004), "Identification of Communities of Peers by Trust and Reputation". En: Proceedings of the 11th International Conference in Artificial Intelligence: Methodology, Systems, and Applications AIMSA 2004, Varna, Bulgaria, Sept. 2-4, 2004, LNCS 3192, Springer, Berlin, pp. 85-95.

[2] Brunkhorst, I., Dhraief, H., Kemper, A., Nedjl W., Wiesner, C. (2003), "Distributed Queries and Query Optimization in Schema-Based P2P Systems". En: Proceedings of the 1st International Workshop on Databases, Information Systems, and Peer-to-Peer Computing DBISP2P, Berlin, Germany, Sept. 7-8, 2003, LNCS 2944, Springer, Berlin, pp. 184-199.

[3] Carrillo-Ramos, A., Gensel, J., Villanova-Oliver, M., Martin, H. (2005), "A Peer Ubiquitous Multi-Agent Framework for providing nomadic users with adapted information". En: Post-proceedings of the 4th International Workshop on Agents and P2P Computing AP2PC 2005, Utrecht, Netherlands, July 26, 2005, LNAI 4118, Springer, Berlin, pp. 159-172.

[4] Carrillo-Ramos, A., Villanova-Oliver, M., Gensel, J., Martin, H. (2007), "Knowledge Management for Adapted Information Retrieval in Ubiquitous Environments". En: Proceedings of the 2nd International Conference on WebIS and Tech WEBIST 2006, Setubal, Portugal, April 11-13, 2005, LNBIP 1, Springer, Berlin, pp.84-96.

[5] Carrillo-Ramos, A., Villanova-Oliver, M., Gensel, J., Martin, H. (2006), "Génération de profils contextuels à partir de préférences d'utilisateurs nomades". En: número especial de "Adaptation et gestion du contexte", Revue Ingénierie des Systèmes d'Information, volume 11, número 5, pp. 61-88.

[6] Carrillo Ramos, A. (2007), "Agents ubiquitaires pour un accès adapté aux systèmes d'information: Le Framework PUMAS". Tesis doctoral de la Universidad Joseph Fourier, Grenoble, Francia, sustentada el 5 de marzo de 2007.

[7] Idreos, S., Tryfonopoulos, C., Koubarakis M., Drougas, Y. (2004), "Query Processing in Super-Peer Networks with Languages based on Information Retrieval: The P2P-DIET Approach". En: Proceedings of the 9th International Conference on Extending Database Technology EDBT 2004, Heraklion, Greece, March 14 - 18, 2004, LNCS 3268, Springer, Berlin, pp. 496-505.

[8] Indulska, J., Robinson, R., Rakotonirainy, A., Henricksen, K. (2003), "Experiences in Using CC/PP in Context-Aware Systems". En: Proceedings of the 4th International Conference on Mobile Data Management MDM 2003, Melbourne, Australia, January 21-24, 2003, LNCS 2574, Springer, Berlin, pp. 247-261.

[9] Kokkinidis, G., Lefteris, S., Christophides, V. (2006), "Query Processing in RDF/S-Based P2P Database Systems". En: Semantic Web and Peer-to-Peer, Springer, Berlin, pp. 59-87.

[10] Koloniari, G., Pitoura, E. (2004), "Content-Based Routing of Path Queries in Peer-to-Peer Systems". En: Proceedings of the 9th International Conference on Extending Database Technology. EDBT 2004, Heraklion, Greece, March 14 - 18, 2004, LNCS 3268, Springer, Berlin, pp. 29-47.

[11] Odell, J., Van Dyke Parunak, H., Bauer, B. (2000), "Representing Agent Interaction Protocols in UML". En: Proceedings of the 1st International Workshop on Agent Oriented Software Engineering AOSE 2000, Limerick, Ireland, June 10, 2000, LNCS 1957, Springer, Berlin, pp. 121-140.

[12] Panti, M., Penserini, L., Spalazzi, L. (2002), "A Multi-Agent System based on the P2P model to Information Integration". En: Proceedings of the 1st International Conference on Autonomous Agents and Multi-Agent Systems AAMAS 2002, Bologna, Italy. 2002, Disponible en: http://www.agentcities.org/EUNET/Projects/acnet_proj_38.pdf (Consultado en Agosto 2007)

[13] Park, J., Barber, S. (2004), "Finding Information Sources for Model Sharing in Open Multi-Agent System". En: Proceedings of the Workshop on Agents for Ubiquitous Computing UbiAgents04, July 20th, 2004, Columbia University, New York City. Disponible en: http://www.ift.ulaval.ca/~mellouli/ubiagents04/ (Consultado en Agosto 2007)

[14] Röhm, U., Böhm, K., Schek, H. (2000), "OLAP Query Routing and Physical Design in a Database Cluster". En: Proceedings of the 7th Conference on Extending Database Technology EDBT 2000, Konstanz, Germany, March 27-31, 2000, LNCS, vol. 1777, Springer, Berlin, pp. 254-268.

[15] Siberski W., Thaden, U. (2004), "A simulation Framework for Schema-Based Query Routing in P2P Network". En: Proceedings of the 9th International Conference on Extending Database Technology EDBT 2004, Heraklion – Crete Greece, March 14 - 18, 2004, LNCS, vol. 3268, Springer, Berlin, pp. 436-445.

[16] Shizuka, M., Ma, J., Lee, J., Miyoshi, Y., Takata, K. (2004), "A P2P Ubiquitous System for Testing Network Programs". En: Proceedings of the Embedded and Ubiquitous Computing EUC 2004, Aizu, Japan, August 25-27, 2004, LNCS 3207, Springer, Berlin, pp. 1004-1013.

[17] Stuckenschmidt, H., Van Harmelen, F., Siberski, W., Staab, S. (2006), "Peer-to-Peer and Semantic Web". En: Semantic Web and Peer-to-Peer. Springer, Berlin, pp. 1-17.

[18] Thilliez M., Delot T. (2004), "Evaluating Location Dependent Queries Using ISLANDS". En: Proceedings of the Symposium on Advanced Distributed Systems ISSADS 2004, Guadalajara, Mexico, January 25-30, 2004, LNCS 3061, Springer, Berlin, pp. 126-136.

[19] Vdovjak, R., Houben, G-J., Stuckenschmidt, H., Aerts, A. (2006), "RDF and Traditional Query Architectures". En: Semantic Web and Peer-to-Peer. Springer, Berlin, 2006, pp. 41-58.

[20] Xu, J., Lim, E., Ng, W.K. (1999), "Cluster-Based Database Selection Techniques for Routing Bibliographic Queries". En: Proceedings of the 10th International Conference on Database and Expert Systems Applications DEXA 99, Florence, Italy, Aug. 30 – Sept. 3, 1999, LNCS 1677, Springer, Berlin, pp.100-109.

[21] Yang, D., Xu, L., Cai, W., Zhou, S., Zhou, A. (2004), "Efficient Query Routing for XML Documents Retrieval in Unstructured Peer to Peer Networks". En: Proceedings of the 6th Asia Pacific Web Conference APWeb 2004, Hangzhou, China, April 14-17, 2004, LNCS 3007, Springer, Berlin, pp. 217-223.

[22] Zhuge, H., Liu, J., Feng, L., He, C. (2004), "Semantic-Based Query Routing and Heterogeneous Data Integration in Peer to Peer Semantic Link Networks". En: Proceedings of the 1st International IFIP Conference on Semantics of a Networked World and Semantics for Grid Databases ICSNW 2004, Paris, France, June 17-19, 2004, LNCS 3226, Springer, Berlin, pp. 91-107.

Angela C. Carrillo Ramos. Ingeniera de Sistemas y Computación de la Universidad de los Andes, Bogotá, Colombia (1996). Magíster en Ingeniería de Sistemas y Computación de la Universidad de los Andes, Bogotá, Colombia (1998). Doctorado en Informática de la Universidad Joseph Fourier, Grenoble, Francia (2007). Asistente de Investigación y Profesora de Cátedra de la Universidad de los Andes (1996-1997). Profesora Asistente de la Universidad de los Andes (1998-2003). Actualmente es Profesora Asociada e Investigadora de los grupos ISTAR y SIDRE de la Pontificia Universidad Javeriana de Bogotá. Su trabajo se ha enfocado en el acceso a sistemas de información a través de dispositivos móviles utilizando la tecnología de agentes. Otros de sus intereses son la adaptación (personalización) de la información en ambientes nómadas de acuerdo al usuario y al contexto y, la construcción de software.

Marlène Villanova-Oliver. Doctorado en Informática del Instituto Nacional Politécnico de Grenoble, Francia (2002). Actualmente se desempeña como Profesora Asistente en la Universidad Pierre Mendes France (UPMF) y el Instituto Universitario de Tecnología (IUT2), del departamento STID de Grenoble, Francia. Además es investigadora en el equipo STEAMER del Laboratorio de Informática de Grenoble (Francia). Sus temas de interés son los sistemas de información multimedia, la adaptación de la información y la representación de conocimiento teniendo en cuenta aspectos espacio-temporales.

Jérôme Gensel. DEA en Informática del Instituto Nacional Politécnico de Grenoble, Francia (1990). Doctorado en Informática de la Universidad Joseph Fourier, Grenoble, Francia (1995). Habilitación para dirigir proyectos de Investigación de la Universidad Joseph Fourier (2006). Actualmente se desempeña como Profesor Asociado en la Universidad Pierre Mendes France (UPMF) y profesor de Informática en la UFR Ciencias del Hombre y de la Sociedad (UFR SHS), en el departamento de Informática y Matemáticas y Ciencias Sociales (IMSS). Además es investigador en el equipo STEAMER del Laboratorio de Informática de Grenoble (Francia). Sus temas de interés son los sistemas de información web, la adaptación de la información, sistemas de información geográficos, representación de conocimiento, movilidad y ubicuidad.

Hervé Martin. Doctorado en Informática de la Universidad Joseph Fourier, Grenoble, Francia (1991). Habilitación para dirigir proyectos de Investigación de la Universidad Joseph Fourier (2000). Actualmente se desempeña como Profesor Asociado en la Universidad Joseph Fourier y en el Instituto de Geografía Alpina en Grenoble, Francia. Director e investigador del equipo STEAMER y director Adjunto del Laboratorio de Informática de Grenoble (Francia). Sus temas de interés son los sistemas de información multimedia pervasivos y los sistemas de información geográficos.