Aplicación de técnicas multicast para la mejora del sistema de descubrimiento de recursos en Grids

Carlos Castillo-Trello y Rafael Moreno-Vozmediano

Resumen— **Este artículo analiza los sistemas de descubrimiento y de monitorización dentro del contexto de sistemas distribuidos heterogéneos y dinámicos como son los entornos Grid, que necesitan arquitecturas que se adapten a las características del tipo de sistema en cuestión. Sin embargo, algunos sistemas como el servicio de monitorización y descubrimiento de Globus (MDS) pueden resultar demasiado rígidos siendo mejorables fácilmente haciendo uso de recursos que brinda la red, como por ejemplo el envio multicast, que permitirá ahorrar ancho de banda y sobrecargar menos los directorios consultados. A pesar de que la especificación de los protocolos multicast es relativamente madura, en este contexto se plantean diversos problemas en cuanto a la fiabilidad y la seguridad que todavía no tienen una solución sencilla ni totalmente definida.**

Palabras clave— **Grid, sistemas de monitorización y descubrimiento, multicast, MDS, Globus-XIO.**

I. Introducción

UN a de las características intrínsecas de cualquier sistema grid es que se trata de un entorno altamente cambiante y dinámico, ya que en cualquier instante se pueden agregar nuevos recursos o servicios, o algunos de los recursos/servicios existentes pueden dejar de estar disponibles o cambiar sus atributos. En este contexto es imprescindible la utilización de un sistema de información que permita descubrir los recursos existentes en la red y controlar los cambios que puedan producirse en éstos. El sistema de información utilizado en la mayoría de entornos grid construidos sobre la plataforma Globus se basa en MDS (Monitoring and Discovery System), que proporciona un conjunto de componentes, servicios e interfaces para el descubrimiento y monitorización de recursos y servicios en el grid.

Una de las componentes principales de MDS es el servidor índice (Index Server) [16] que se ejecuta en cada recurso y recopila información acerca de las propiedades y servicios disponibles en el mismo, mediante el acceso a distintos proveedores de información. Estos servidores índices se pueden organizar de forma jerárquica, de manera que un servidor índice de una organización puede recopilar, agregar e indexar información de los distintos servidores locales de cada recurso, creando un directorio de recursos y servicios. Esta organización es totalmente flexible, pudiendo existir varios niveles de servidores índices en la jerarquía o distintos servidores que indexan y agregan distintos tipos de información. Los usuarios del grid pueden realizar consultas a estos servidores para descubrir los recursos de una red o una organización. Asimismo, un usuario también puede realizar una consulta al servidor índice local de un recurso concreto para obtener información concreta y actualizada del mismo.

Esta organización, basada en servidores que agregan e indexan información sobre los recursos disponibles en la red presenta varios inconvenientes. Por una parte, presenta un punto central de fallo, ya que si el servidor índice que contiene la información de todos los recursos falla, no será posible descubrir los recursos disponibles en la red. Por otra parte, la actualización de la información mantenida en el servidor se gestiona mediante tiempos de vida. En entornos muy cambiantes, puede ocurrir que la información suministrada por el servidor sea obsoleta y no se corresponda con la situación real de la red.

En este artículo se propone una organización alternativa del sistema de descubrimiento de recursos basada en multicast. Este sistema no necesita disponer de un servidor central que agregue e indexe información, sino que el usuario accede directamente a todos los servidores índice locales mediante una consulta multicast. De esta forma se elimina el punto central de fallo y, por otra parte, puesto que la información se obtiene directamente del servidor índice local del recurso, ésta estará más actualizada. Para desarrollar este sistema se han implementado un mecanismo de tipo *peer-to-peer*, basado en dos agentes: el agente de usuario (UA, *user agent*) y el agente del recurso (RA, *resource agent*). Estos agentes se ejecutan respectivamente en el sistema del usuario y en el sistema que contiene los recursos/servicios que se quieren descubrir y la comunicación entre ambos se realiza mediante multicast.

Este artículo tiene la siguiente estructura: En la sección II se establece una clasificación de los distintos tipos de arquitecturas de descubrimiento de recursos/servicios, estudiando las ventajas e inconvenientes que presentan cada uno de estos sistemas. En la sección III se estudia el sistema de monitorización y descubrimiento utilizado en Globus: MDS. Se definen las características principales de este sistema y como encaja dentro de la clasificación realizada, quedando por tanto claras sus ventajas y desventajas. En la sección IV, se analiza una propuesta de arquitectura mejorada utilizando multicast. Se verán las ventajas que introduce esta arquitectura y por otro lado se verán los problemas que introduce. Finalmente la sección V, resume las conclusiones a las

Departamento de Arquitectura de Computadores y Automática. Grupo de Investigación de Arquitectura de Sistemas Distribuidos (ASD). Universidad Complutense de Madrid: ccastillo@fdi.ucm.es, rmoreno@dacya.ucm.es

que hemos llegado con este trabajo y se definen unas líneas de futuro trabajo que tratarán de mejorar la propuesta definiendo una arquitectura nueva y las herramientas y útiles necesarios para conseguirlo.

II. Arquitecturas de descubrimiento de recursos/servicios

Los mecanismos de descubrimiento de recursos o servicios existentes se pueden clasificar en dos grandes categorías:

A. Arquitecturas Peer-to-Peer

Las arquitecturas *peer-to-peer* (P2P) utilizan mecanismos completamente distribuidos para el descubrimiento de los servicios o recursos, en los que las entidades de red (proveedores o clientes) negocian uno a uno entre ellos con el fin de encontrar los servicios disponibles y sus atributos, y que cumplan además una serie de requisitos impuestos por el usuario. Se pueden utilizar dos mecanismos básicos para el descubrimiento de servicios o recursos en los sistemans peer-to-peer [1]: mecanismos basados en consultas (*queries*) y mecanismos basados en anuncios (*advertisements*).

A.1 Sistemas P2P basados en consultas (P2P-Query)

En los sistemas *P2P-Query*, también conocidos como mecanismos P2P activos o de *pull*, los clientes envían un mensaje de descubrimiento a la red, mediante broadcast o multicast, preguntando por aquellos servicios o recursos que cumplan unos determinados requisitos o atributos. Los proveedores responden a la petición del cliente enviandando una descripción de los atributos del servicio o del recurso. Ejemplos de sistemas P2P basados en consultas son el Service Discovery Protocol (SDP) que se utiliza en Bluetooth [2], la propuesta de mecanismo de descubrimiento de servicios para redes ad-hoc bajo demanda [3][4], y el protocolo Konark activo o *pull* [5].

A.2 Sistemas P2P basados en anuncios (P2P-Adv)

En los sistemas *P2P-Adv*, conocidos también como pasivos o mecanismos *push*, los proveedores anuncian periódicamente, mediante broadcast o multicast la localización y los atributos de los recursos y de los servicios, de forma que los clientes pueden construir una base de datos local con todos los recursos disponibles en la red. Ejemplos de sistemas P2P basados en anuncios el servicio de descubrimiento son Universal Plug and Play (UPnP) [6] desarrollado por Microsoft, y el protocolo Konrak pasivo o de *push* [5].

En general, las arquitecturas *peer-to-peer* son útiles en entornos muy dinámicos, donde la infraestructura de la red es impredecible, y la presencia de directorios dedicados no está garantizada en todo momento. Sin embargo, estos mecanismos, especialmente los de tipo *P2P-Adv*, que estn basados en broadcasting (inundación o *flooding*) o bien multicast, hacen un uso enorme de ancho de banda y tienen una escalabilidad muy reducida. Por lo tanto solamente son apropiados en redes de reducido tamaño. Este desperdicio de recursos se debe al envío de información de forma periodica que no ha sido solicitada. Sin embargo, se reduce el tiempo de búsqueda o *lookup*, ya que todos los clientes disponen de información actualizada sobre todos los recursos y servicios que están disponibles en la red.

B. Arquitecturas basadas en directorio

Las arquitecturas de descubrimiento basadas en directorio utilizan un repositorio centralizado o distribuido que agrega e indexa la información obtenida sobre los recursos y servicios disponibles en la red. Los proveedores registran los recursos y servicios en este directorio y los clientes relizan consultas sobre este directorio para obtener la información sobre los recursos o servicios. Hay tres esquemas generales distintos de sistemas basados en directorio: directorio centralizado, directorio plano distribuido y directorio jerárquico distribuido.

B.1 Arquitectura de directorio centralizado (CD)

. La arquitectura CD (Central Directory) está basada en un directorio central que agraga la información de todos los proveedores, y responde las consultas de todos los clientes. La arquitectura de directorio centralizado es una solución sencilla, fácil de administrar. Sin embargo, el directorio puede representar un cuello de botella y a la vez un punto de fallo crítico, que podría ocasionar el fallo de todo el sistema. Se pone de manifiesto que esta solución no escala bien y es únicamente apropiada para redes pequeñas. Algunos ejemplos de mecanismos de descubrimiento basados en arquitecturas centralizadas son Service Location Protocol (SLP) [7] estandarizado por Internet Engineering Task Force (IETF, RFC 2608), el sistema Jini [8], que es una servicio de descubrimiento independiente de la plataforma, basado en Java y desarrollado por Sun Microsystems, y la propuesta basada en agentes, Agent-Based Service Discovery, descrita en [9].

B.2 Arquitecturas de directorio distribuido plano (DFD)

En la arquitectura DFD (Distributed Flat Directory) varios directorios cooperan de forma *peer-to-peer*, para mantener un repositorio distribuido de información sobre recursos y servicios. Los directorios distribuidos planos pueden trabajar de dos formas distintas. Los directorios pueden intercambiar información con todos los restantes directorios, normalmente mediante multicast, de forma que en cada directorio se mantenga una base de datos completa sobre todos los recursos y servicios de la red. Dos ejemplos de mecanismos de descubrimiento de este tipo son INS (Intentional Naming Service) [10] y el protocolo de saludo (Salutation) [11]. Es evidente que esta solución genera un gran volumen de tráfico, y que por lo tanto no es escalable. La segunda alternativa es dividir la red en clusters o dominios, de forma que cada directorio mantenga un repositorio

con información sobre los servicios y recursos dentro de un cluster o dominio dado. El intercambio de información entre los directorios que estan en diferentes clusters se puede conseguir utilizando un esquema peer-to-peer, pero con una frecuencia de anuncios más baja que la frecuencia dentro del cluster, como por ejemplo el sistema INS/Twine [12], o se puede conseguir bajo demanda, como por ejemplo el sistema de localización de servicios basado en Virtual Backbone [13]. Aunque las soluciones con clusters son más escalables y adecuadas para redes de gran tamaño, para poder gestionar los clusters se deben utilizar algoritmos complejos (en tareas como la formación del cluster, selección de directorios, añadir o eliminar nodos al cluster, etc.) y además garantizar la estabilidad del cluster.

B.3 Arquitecturas de directorio distribuido jerárquico (DHD)

En la arquitectura DHD (Distributed Hierarchical Directory), la red se divide en dominios con una estructura jerárquica (como el servicio de DNS) y los directorios tiene relaciones padre e hijo. Esta solución es altamente escalable, pero requiere una organización de red jerárquica y por tanto muy rígida, que no es apropiada en entornos dinámicos. Algunos ejemplos de arquitecturas de directorio distribuido jerárquico son el servicio de monitorización y descubrimiento (MDS) [14][15], y el Secure Service Discovery Service (SDS) desarrollado en UC Berkeley [17].

TABLA I
Clasificación de los mecanismos de descubrimiento de servicios y recursos.

Arquitecturas Peer-to-Peer (P2P)	Arquitecturas basadas en directorio
Mecanismos P2P Query (activos, pull)	Directorio centralizado
	Directorio Distribuido plano
Mecanismos P2P basados en anuncios (pasivos, push)	Directorio Distribuido jerarquico

. En la tabla I tenemos de forma resumida esta clasificación y la tabla II.muestra las principales características de cada uno de estos esquemas.

III. El servicio MDS de Globus

El sistema MDS (Monitoring and Discovery System) [15] es un conjunto de servicios web [18] que permite monitorizar y descubrir los servicios y recursos de Grids. A diferencia de su predecesor, MDS2 [14], ya no se basa en la arquitectura LDAP, sino que sigue las especificaciones de WSRF. Este sistema permite a los usuarios descubrir qué recursos son considerados parte de una Organización Virtual (VO, o Virtual Organization) y monitorizar estos recursos. Podemos ver una descripción detallada del sistema MDS en [16]. Los servicios de MDS facilitan una serie de interfaces para realizar consultas, suscripciones a determinados detalles de información y sistemas de *trigger* que se pueden configurar para que se realice una acción al producirse una determinada situación (por ejemplo mandar un email al administrador del sistema cuando el espacio de disco sea inferior al 10%). Los servicios de la implementación actual de WS MDS (o MDS4), que es la versión suministrada con Globus Toolkit 4, obtienen la información mediante un interfaz fácilmente extensible que permite:

- realizar consultas a los servicios WSRF para obtener información de las propiedad de los recursos
- ejecutar un programa para obtener datos
- interactuar con sistemas de monitorización diferentes a MDS

TABLA II
Principales características de los mecanismos de descubrimiento.

	P2P-Query	P2P-Adv	CD	DFD	DHD
Grado de dinamismo	Alto	Alto	Bajo	Bajo	Bajo
Escalable	Bajo	Bajo	Bajo	Alto	Alto
Consumo de ancho de banda	Medio	Alto	Bajo	Bajo	Bajo
Retardo de descubrimiento	Alto	Bajo	Bajo	Bajo	Bajo
Complejidad de gestión	Bajo	Bajo	Medio	Alto	Alto

Los recursos de Grid y sus servicios pueden *anunciar* una gran cantidad de información que puede ser útil para distintos casos de uso. MDS4 fue diseñado específicamente para resolver las necesidades de un sistema de monitorización en Grid —es decir un sistema que publica datos de forma que estén dispoble para un gran número de personas y de dispositivos. Por lo tanto, no se trata de un sistema que maneje eventos, como NetLogger, o un monitor propiamente dicho de un cluster, sino que puede acceder a sistemas y archivos de monitorización más detallados y puede publicar un resumen de dicha información siguiendo unos interfaces estándar.

Podemos ver de forma simplificada la arquitectura de MDS en la figura 1.

Resumiendo podemos decir que los clientes MDS pueden utilizar un mecanismo de suscripción de forma que sean notificados cuando se produzca algún tipo de cambio en la información. MDS soportan

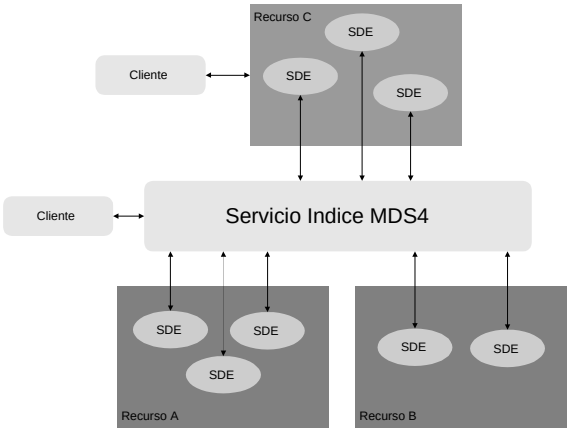

Fig. 1. Arquitectura de MDS4

consultas realizadas por el cliente que pueden ser de distinto grado de complejidad. Se puede realizar "Consultas por nombre" especificando el nombre de un servicio y uno o más nombres de sus *Service Data Elements*. Opcionalmente los clientes pueden indicar una expresión de tipo `XPath` para seleccionar de forma más precisa la información que desean obtener de la consulta. Podemos por tanto obtener información mediante el tipo de recurso y por comparación entre los valores de los recursos, por ejemplo `CPULoad` mayor que un determinado valor. Mediante el servicio `Index Service` podemos filtrar o fusionar datos de distintas fuentes.

Como fuentes de información los servidores de índices pueden utilizar distintos proveedores, entre los que cabe destacar:

- Información de Hawkeye y Ganglia, como nombre e id del host, información sobre el procesador, cantidad de memoria, nombre del Sistema Operativo y versión, tipo de sistema de archivos, carga del procesador, ...
- *WS GRAM*, el servicio de Globus que permite enviar trabajos, publica información relativa a: información de la cola de trabajos, número de CPUs disponibles, número de trabajos, estadística sobre uso de memoria.
- *Reliable File Transfer Service (RFT)*, el servicio de transferencia de archivos de Globus. Este servicio de tipo WSRF publica la siguiente información: estado de los datos del servidor, estado de la transferencia de un archivo o de un conjunto de archivos, número de transferencias activas, información sobre los recursos utilizados al ejecutar el servicio.
- *Community Authorization Service (CAS)*, este servicio identifica al VO al que da servicio de autenticación.
- Cualquier otro servicio WSRF que publique información. En general todos los servicios GT4 publican un elemento `ServiceMetaDataInfo` que incluye el instante de inicio el servicio, la versión y el nombre del tipo de servicio.

IV. Propuesta de mejora mediante multicast y arquitecturas alternativas

Como hemos visto en la sección anterior MDS4 permite crear un sistema de información para la realización de consultas y suscripción a determinadas fuentes de información. La estructura del directorio es jerarquica al tener que definir Indices que se encarguen de recoger información que es parte de un filtrado o mezcla de otras directorios MDS, para poder tener una visin global del sistema grid. Esto genera varios tipos de problema. El servidor índice que tiene la información de todos los recursos es un punto central de fallo, ya que si deja de estar disponible, no será posible descubrir los recursos de la red. Por otro lado, tenemos el problema de la obtención de información actual. Al actualizarse la información mediante tiempos de vida, si el sistema es suficientemente grande, obtendremos datos obsoletos. Otro es que no permite recibir información de cierto tipo a un grupo de clientes, lo cual evita tener que repetir esa consulta varias veces. En entornos muy dinámicos, en los que existe una gran variabilidad de los elementos que pertenecen al Grid, se deben resolver este tipo de incoveniente para poder lograr una mayor fiabilidad.

El sistema propuesto permite resolver estos problemas al realizar las consultas mediante multicast, en lugar de consultar a un servidor de índices principal. Además, como se suministra la información simultáneamente a un grupo de usuarios, evita un número excesivo de consultas a los servidores y reduce el consumo de ancho de banda. La arquitectura, que se muestra en la figura 2, consiste en un usuario o grupo de usuarios que quiere obtener una información determinada de un grupo de máquinas. El procedimiento es el siguiente:

1. En cada uno de los nodos en los que reside un sistema de directorio MDS instalamos un agente del recurso (*RA*, resource agent). Este agente actua como servidor, esperando peticiones de un agente cliente por la dirección multicast `228.5.6.7` y puerto `6789`.
2. Un cliente realizará la petición mediante el agente de usuario (*UA*, user agent), que enviará el nombre del índice de MDS. La implementación utiliza un socket multicast con la dirección y puerto que acabamos de indicar (`228.5.6.7` y `6789`).
3. Los *RA*, recibirán el nombre de índice a consultar. Ejecutarán localmente la órden `wsrf` en el servidor MDS local.
4. La información proporcionada por MDS será enviada por un socket multicast a la dirección `228.5.6.8` y puerto `6790`
5. Todos los *UA* se habrán quedado a la espera de recibir por la dirección `228.5.6.8` y puerto `6790` la información solicitada. Si en un tiempo de $30s$. no reciben ningún paquete, suponemos que no hay respuesta por parte de los agentes. Este *time-out* debe ser ajustado dependiendo de las

condiciones da carga de la red y de las máquinas donde reside el sistema MDS.

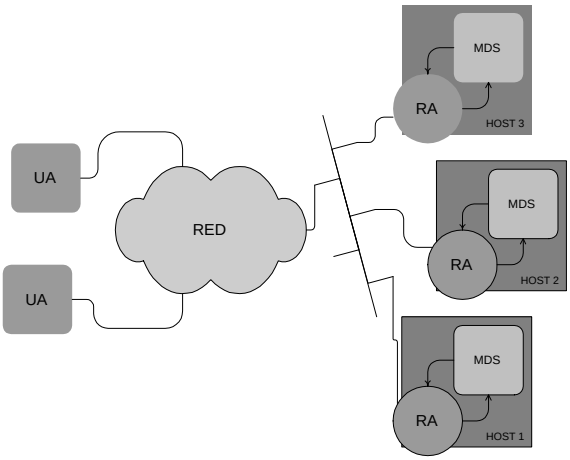

Fig. 2. Arquitectura para permitir realizar consultas a multiples directorios MDS

Se han realizado diversas medidas en el entorno de laboratorio, siendo por lo tanto un entorno de pruebas LAN de tamaño reducido. Se utilizaron 2, 3 y 4 nodos midiendo la cantidad de información en transito debido a los paquetes multicast, retardos y el ancho de banda consumido. Para simular un entorno de producción se han realizado estas medidas con distinto niveles de carga (bajo y medio) tanto de la red como de las máquinas. Las tablas III y IV presentan estos resultados de forma esquemática.

TABLA III
MEDIDAS REALIZADAS SOBRE EL ENTORNO DE PRUEBAS CON UN NIVEL DE CARGA BAJO.

Núm. nodos	Tamaño (bytes)	Retardo (ms.)	Ancho de banda (Kbps)
2	613	500	445
3	3628	1000	778
4	9084	1250	738

TABLA IV
MEDIDAS REALIZADAS SOBRE EL ENTORNO DE PRUEBAS CON UN NIVEL DE CARGA MEDIO.

Núm. nodos	Tamaño (bytes)	Retardo (ms.)	Ancho de banda (Kbps)
2	613	1000	408
3	3628	2000	689
4	9084	2500	667

Si se realizaran estas consultas sin esta implementación multicast, se necesita hacer primero una consulta al servidor de índices central. Después se parsea esta información para saber los recursos que componen el grid. A continuacin para disponer de la información actualizada, se debe consultar los sistemas MDS de cada host uno a uno. El tiempo de retardo global por lo tanto aumentaría. Al realizar las medidas en el laboratorio para 4 nodos, con un nivel de carga media, se obtiene un retardo alrededor de los 5000 ms, más el retardo de la consulta inicial al índice central de $\sim$ 2500 ms. Si se hacen las consultas concurrentemente, se reduce el tiempo total a 2500 ms de la consulta inicial más 2500 ms de la consulta más lenta. Sin embargo este método supone aumentar el ancho de banda consumido, sobre todo en determinados instantes de tiempo.

V. Conclusiones y trabajos futuros

La arquitectura multicast propuesta resuelve los problemas de la existencia de un punto central de fallo, ya que no es necesario un sistema central de información para conocer los recursos que componen el grid. Por otro lado, la información obtenida con este método es la más reciente, al ser cada uno de los directorios locales los que responden directamente a las peticiones. De esta forma se reduce el retardo global, al eliminar esta consulta inicial al servidor central y al obtener la información de los RA de forma simultánea.

La principal limitación de esta implementación es que se usan agentes de recursos en cada uno de los nodos que realiza las consultas en el sistema MDS local. Por lo tanto no estamos siguiendo el sistema de seguridad definido por Globus. En una futura implementación, se tratará de integrar este sistema con el código de Globus. Se deberá seguir, tanto en el UA como en los RA, el servicio de seguridad de Globus de delegación [20]. Se utilizarán los mecanismos básicos GSI [21] junto con las últimas especificaciones de seguridad propuestas para implementar los estándares de seguridad en servicios web [22]. Sin embargo, sobre seguridad queda un gran camino por recorrer, ya que los mecanismos de autenticación para grupos es un tema cuya estandarización todavía está en proceso de discusión.

Otra posible optimización sería utilizar la librería Globus XIO para implementar una modificación del sistema MDS cargando una pila de protocolos, que incluye un protocolo multicast. El principal problema es que los servicios multicast soportados por la red son UDP (no orientados a conexión y no fiables), y por este motivo no es directamente implementable con los servicios multicast que brinda Globus. Sin embargo existen trabajos que han implementado un servicio de transporte multicast fiable (semejante a TCP) utilizando UDP multicast [23], que se ha integrado dentro de la arquitectura XIO, permitiendo disponer de este servicio de forma sencilla.

En un futuro se realizarán medidas en un entorno Grid con hosts conectados mediante una WAN, para poder estudiar el comportamiento del sistema. Esto será posible gracias a la Red Telemática de Investigación de Alta Velocidad, conocida como REDI-Madrid [24]

Referencias

[1] Rafael Moreno-Vozmediano, Resource Discovery in Ad-Hoc Grids, International Conference on Computational

Science (4) 2006: pp. 1031-1038
[2] Bluetooth, Service Discovery Protocol, Bluetooth Specification Version 1.1, Part E, Feb. 2001
[3] R. Koodli, C. Perkins, Service discovery in on-demand ad hoc networks, IETF Internet Draft (draft-koodli-manet-servicediscovery-00.txt), October 2002.
[4] Christian Frank, Holger Karl, Consistency challenges of service discovery in mobile ad hoc networks, Proc. of the 7th ACM Int. symposium on modelling, analysis and simulation of wireless and mobile systems (MSWiM), 2004, pp. 105-114.
[5] S. Helal, N. Desai, V. Verma, C. Lee, *Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks*, Proc.of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003
[6] B.A. Miller, T. Nixon, C. Tai, M. D. Wood, Home Networking with Universal Plug and Play, IEEE Communications Magazine, vol. 39, no. 12 (2001), pp. 104-109.
[7] E. Guttman, Service Location Protocol: Automatic Discovery of IP Network Services, IEEE Internet Computing. Vol. 3, No. 4 (1999), pp. 71-80
[8] Sun Microsystems, Jini. Architecture Specification, Technical Report, version 1.2, 2001.
[9] Y. Lu , A. Karmouch, M.Ahmed, R.Impey, Agent-Based Service Discovery in Ad-Hoc Networks, 22nd B. Symp. on Communications, Kingston, Ontario, Canada, 2004.
[10] W. Adjie-Winoto, E. Schwartz, H. Balakrshnan, J. Lilley, The design and implementation of an intentional naming system, Proc. of the 17th ACM Symposium on Operating Systems Principles, 34(5), 1999, pp. 186-201.
[11] The Salutation Consortium, Salutation Architecture Specification, Technical Report, version 2.0c, 1999.
[12] M. Balazinska, H. Balakrishnan, D. Karger, INS/Twine: A Scalable Peer-to-Peer architecture for Intentional Resource Discovery, Int.Conf. on Pervasive Computing, Zurich, Switzerland, 2002.
[13] J. Liu, Q. Zhang, W. Zhu, B. Li, Service Locating for Large-Scale Mobile Ad Hoc Network, International Journal of Wireless Information Networks, Vol. 10, No. 1(2003), pp. 33-40.
[14] The Globus Alliance. Globus Toolkit 2.2 MDS Technology Brief, Draft 4, http://www.globus.org/toolkit/docs/2.4/mds/mdstechnologybriefdraft4.pdf, 2003.
[15] The Globus Alliance, MDS Functionality in GT4. WSRF Technical Resources, http://www.globus.org/toolkit/docs/4.0/info/key-index.html, 2005.
[16] J. M. Schopf, M. D'Arcy, N. Miller, L. Pearlman, I. Foster, C. Kesselman, Monitoring and Discovery in a Web Services Framework: Functionality and Performance of the Globus Toolkit's MDS4, Preprint ANL/MCS-P1248-0405, April 2005
[17] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, R. Katz, An Architecture for a secure Service Discovery Service, Proc. of the ACM/IEEE MOBICOM, 1999, pp. 24-35.
[18] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, Web Services Architecture, http://www.w3.org/TR/2003/WD-ws-arch-20030808/ W3C, Working Draft, 2003
[19] The Globus Alliance, Globus - XIO, http://www.globus.org/toolkit/docs/4.0/common/xio/
[20] The Globus Alliance, Security: Delegation Service, http://www.globus.org/toolkit/docs/4.0/security/delegation/
[21] The Globus Alliance, GSI-OpenSSH, http://www.globus.org/toolkit/docs/4.0/security/openssh/
[22] The Globus Alliance, SweGrid Accounting System (SGAS), http://www.globus.org/toolkit/docs/4.0/techpreview/sgas/index.html
[23] K. Jeacle, Jon Crowcroft, A Multicast Transport Driver for Globus XIO, WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005, pp. 284–289
[24] REDIMadrid, Red Telemática de Investigación de Alta Velocidad, http://www.madrimasd.org/queesmadrimasd/RedTelematicaMadrid/default.asp