Informe sobre Sistemas de Computación en Redes P2P

EHU-KAT-IK-04-09
Carlos Pérez-Miguel, José Miguel-Alonso y Alexander Mendiburu

Dpto. de Arquitectura y Tecnología de Computadores
UPV/EHU

1 Introducción

La reciente aparición de aplicaciones que explotan el concepto de red Peer-to-Peer o P2P en la que todos sus integrantes poseen el mismo rol ha hecho plantearse sus posibles aplicaciones al mundo de la computación distribuida. Aunque ya existen soluciones parecidas en este campo, como aquellas que siguen el paradigma del Grid Computing, este nuevo concepto puede ser el origen de sistemas de computo masivos totalmente abiertos que cualquier persona con una conexión a Internet podría usar para su beneficio o para el de los demás, compartiendo sus recursos excedentes.

En este informe intentamos dar una visión del estado del arte en cuanto a redes P2P y especialmente en el ámbito de los sistemas de computación distribuidas junto con varias aportaciones que permitirían construir sistemas de High Throughput Computing.

Este informe se desarrolla de la siguiente manera: En la Sección 1 se da una visión general del documento. En la Sección 2 se repasan el concepto de red Peer-to-Peer y los distintos tipos de redes P2P que existen. En la Sección 3 se muestran algunas propuestas de topologías para redes P2P estructuradas. En la Sección 4 se aborda el problema de los sistemas de computación distribuidos mediante redes P2P, las características que debieran tener y que no tienen los sistemas actuales, y se proponen métodos para solventar estas ausencias. En la Sección 5 se presentan algunas herramientas que pudieran ser útiles para el estudio de las redes P2P. En la Sección 6 se describe un plan de trabajo posible para construir un sistema de High Throughput Computing sobre redes P2P y por último en la Sección 7 se muestran las conclusiones.

2 Redes Peer-to-Peer

Los sistemas P2P son sistemas distribuidos en los cuales no existen ningún tipo de control central ni de estructura jerárquica, pudiendo por ello ir más allá de los servicios ofrecidos por los tradicionales sistemas centralizados de tipo Cliente-Servidor a costa de una mayor complejidad técnica y teórica. En los sistemas P2P, los nodos que conforman el sistema se conectan entre si usando redes IP formando una especie de red virtual a nivel de aplicación, también llamada

"overlay", que es usada para enrutar mensajes entre los nodos con el objetivo de buscar información, recursos compartidos, usuarios, etc...

A diferencia de los sistemas de Grid Computing, las redes P2P no se basan en la interconexión de grupos organizados, unidos entre si por redes más o menos fiables. A partir de un gran número de usuarios no fiables con roles similares unidos entre si por conexiones no fiables son capaces de ofrecer una serie de características interesantes en un sistema distribuido, como por ejemplo, escalabilidad, resistencia a fallos, búsqueda eficiente de información, almacenamiento redundante, permanencia o anónimidad.

Podemos considerar un sistema P2P como una arquitectura de varias capas, a saber:

- El nivel de red, representa el nivel más bajo de la arquitectura, ofreciendo las capacidades básicas de comunicación entre ordenadores, bien a través de redes IP o mediante redes Ad-Hoc.
- El nivel de administración de la capa P2P, encargado de enrutar mensajes y de realizar tareas de mantenimiento del *overlay*.
- El nivel de características, que da soporte a funcionalidades de la red como la seguridad, la resistencia a fallos o la administración de recursos.
- El nivel de servicios, que es el encargado de proporcionar funcionalidades al nivel de aplicación.

Según en que capa de la arquitectura nos fijemos podemos considerar diferentes clases de sistemas P2P, sin embargo, la clasificación más básica se refiere a la forma de crear el *overlay* que interconecta los nodos. Según [1], existen dos tipos de *overlays*, los estructurados y los no estructurados.

2.1 Overlays No Estructurados

Una red P2P de tipo no estructurado puede considerarse como un sistema compuesto por nodos interconectados entre si de una forma más o menos aleatoria, sin ningún tipo de conocimiento previo acerca de la topología de la red. La red usa un mecanismo de *flooding* o inundación para comunicar nodos no conocidos entre si a través del overlay. Cuando un nodo desea realizar una consulta, envía mediante broadcast un mensaje con la consulta a sus vecinos, o a parte de ellos según el algoritmo utilizado, que a su vez reenviarán el mensaje al resto de la red mientras que el ámbito del mensaje no supere un límite determinado (de saltos en la red, de tiempo...).

Podemos ver un ejemplo de este mecanismo en la Figura 1, en la cual se muestra una red formada por 6 nodos. En el ejemplo podemos ver como el nodo n_0 desea comunicarse con el nodo n_6, para ello envía mediante broadcast un mensaje de búsqueda a todos sus vecinos, los nodos 1 a 4, en el paso 0; estos reenvían el mensaje en el paso 1 alcanzando el nodo n_6. Podemos observar la poca eficiencia del algoritmo en cuanto a número de mensajes utilizados ya que en el paso 2 n_6 vuelve a recibir el mensaje de búsqueda de n_0 por una ruta distinta.

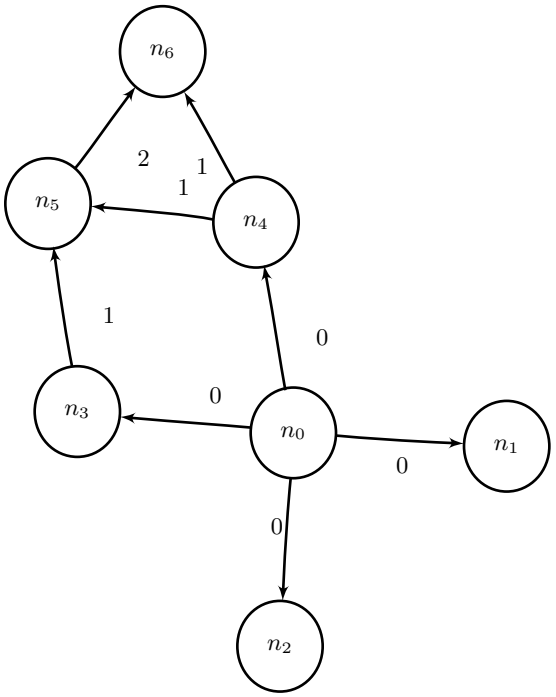

Fig. 1. Sistema de "flooding" en una red P2P.

Mientras que las técnicas basadas en flooding son útiles para encontrar contenidos populares en la red, es decir, altamente replicados, y son inmunes a cambios en cuanto a la composición de la red, tienen una serie de fallos importantes: no son aptos para buscar contenidos raros, es decir, poco replicados, y la carga de trabajo en cada nodo crece linealmente con el número de consultas y de nodos en el sistema. Con lo que, claramente, podemos ver que no es un sistema que escale bien, llegando a saturar los nodos rápidamente si en un instante dado de tiempo el número de estos que conforman el sistema aumenta de forma considerable.

Para evitar este tipo de problemas, se pueden usar otras técnicas de enrutado que son ampliamente utilizadas en las redes Ad-Hoc como las redes de sensores, las cuales se comportan básicamente como una red P2P. Entre dichas técnicas podemos nombrar el enrutado basado en tablas, como por ejemplo "Destination-Sequenced Distance-Vector Routing" [2] o "Clusterhead Gateway Switch Routing" [3].

El inconveniente que tienen estos algoritmos de enrutado es que requieren de tablas de enrutado actualizadas, con la sobrecarga que ello implica en cuanto a tráfico en la red, y dado que en un entorno como el de las redes P2P los integrantes de la misma cambian muy rápidamente, en lugar de este tipo de algoritmos se usan enrutados dinámicos que buscan el mejor enrutado posible bajo

demanda como por ejemplo "Ad Hoc On-Demand Distance Vector Routing" [4], Temporally Ordered Routing Algorithm [5] o Associativity-Based Routing [6]. Estos algoritmos y otros son explicados en [7].

En 1999, Napster [8] hizo popular el concepto de compartición de archivos mediante redes Peer-to-Peer. Fue el primer sistema en percibir que el acceso a contenido popular no debía realizarse a través de un servidor central, si no interconectando los clientes que tuvieran el contenido demandado con otros que no lo tuvieran, formando de esta forma un sistema escalable en el que cuantos más nodos se conectaran al sistema, la capacidad agregada de descarga aumentaría de la misma forma.

Napster basaba su funcionamiento en un directorio centralizado en un servidor que contenía la información sobre qué archivos estaban siendo compartidos por los clientes. En el caso de que algún nodo estuviese interesado en cierto archivo, era el mismo el encargado de interconectarse con el dueño del archivo para su intercambio, es decir, la búsqueda esta centralizada pero no así la descarga de archivos. Llegó a hacerse muy popular hasta que la Asociación de Discográficas Estadounidense (RIAA) demandó a su creador y todo el sistema tuvo que cerrar. Sin embargo la idea de Napster sirvió para crear Gnutella [9–11].

Gnutella es un sistema descentralizado que distribuye tanto la búsqueda como la descarga de archivos al crear un overlay no estructurado entre sus nodos integrantes. Este sistema utiliza el sistema de "flooding" anteriormente explicado aunque desde su creación han surgido una serie de mejoras para evitar el problema del escalado comentado, como por ejemplo, la utilización de supernodos concentradores encargados de agrupar al resto de los nodos facilitando así la búsqueda de información o el uso de WebCaches con el fin de mantener listas actualizadas de nodos conectados al sistema.

2.2 Overlays Estructurados

Cuando hablamos de overlays estructurados, nos referimos a overlays en los que la topología de la red y el contenido son controladas para situarlas en ubicaciones no aleatorias, si no en posiciones del overlay que faciliten su búsqueda. Estos sistemas P2P estructurados basan su funcionamiento en la implementación de una Tabla Hash Distribuida (DHT), en la cual la información es almacenada de forma determinista en el nodo o nodos cuyo identificador corresponda con la clave del objeto a almacenar, siendo dicha clave única en el sistema. En un sistema de DHT se considera un espacio de claves de una longitud determinada y se mapea dicho espacio de claves en la red de nodos que forman la red P2P. Dicho mapeo consiste en asignar a cada uno de los nodos un identificador perteneciente al espacio de claves para luego dividir el espacio de claves en N fragmentos, asignando cada fragmento al nodo cuyo identificador este contenido en dicho fragmento. De esa forma, cada clave del DHT será mantenida por un sólo nodo en la red.

El uso de este DHT nos permite crear una red que soporte el almacenamiento escalable de información en la forma de pares {clave, valor} pudiendo definir una interfaz de uso similar a la de cualquier tabla hash tal y como podemos ver en la

Figura 2 extraída de [1]. Dicha interfaz nos proporciona una serie de funciones para acceder a la información a través de una clave, es decir, como funciones que permitan la inserción/modificación de claves *(put(key, value))* o el acceso a las mismas *(value=get(key))*. Estas funciones implican el uso de un cierto protocolo de enrutado para hacer llegar dichas peticiones al nodo correspondiente a dicha clave.

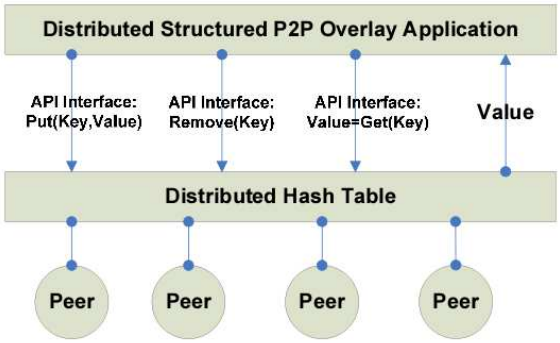

Fig. 2. Interfaz de una red P2P estructurada basada en DHT.

El protocolo de enrutado varía en cada sistema, pero todos ellos cumplen una característica en cuanto a la forma de enrutar un mensaje hacia un nodo destino: el enrutado se realiza de forma progresiva en función de la distancia al destino. Cada nodo del overlay posee una tabla de enrutado en la cual almacena la información acerca del identificador de cada uno de sus vecinos junto con sus direcciones IP. De esta forma, cuando un nodo desea enviar, o enrutar, un cierto mensaje a un nodo identificado por la clave k, enviará el mensaje a aquel de entre sus vecinos que más cerca se encuentre al destino del mensaje. Dicha propiedad de cercanía entre un nodo y una clave del DHT varía entre sistemas, ya que dependerá de la forma de organizar el espacio de claves y de la estrategia de enrutado. En teoría, los sistemas basados en DHTs garantizan que cualquier objeto puede ser localizado en un numero de saltos de orden $O(logN)$ de media, siendo N el número de nodos que forman el sistema. Uno de los puntos débiles de los sistemas basados en DHTs es su comportamiento ante el ingreso y salida masiva de nodos en la red en un momento dado. La latencia puede variar de forma considerable en estos casos y es por ello que se han realizado propuestas para minimizar dichos efectos, como en [12] donde se muestra un algoritmo para alcanzar la latencia casi óptima en grafos que muestran leyes de potencial, como las redes P2P que nos ocupan, y al mismo tiempo conservar las propiedades de enrutado escalable que poseen los DHTs; o como en [13] donde se especifica una solución para mantener un cierto balanceo de carga bajo condiciones adversas. Tambien en [14] se muestra como evitar los problemas derivados de cambios en el sistema eligiendo el subconjunto apropiado de nodos con los que trabajar.

Los protocolos basados en DHTs son buenos buscando información rara, poco demandada. Además de esto, recientemente se han realizado esfuerzos de cara a unificar los sistemas P2P basados en DHTs con objeto de ofrecer una interfaz unificada del enrutado basado en claves (Key Based Routing o KBR) como la Common API [15] de la cual ya existen algunas implementaciones como OpenDHT [16]. La implementación de esta API o alguna equivalente permitiría centrar la investigación bien en temas más topológicos, como el estudio de diversas implementaciones de DHTs y su repercusión en las propiedades de la red P2P, o bien en temas más funcionales, como replicación, seguridad, comunicación a grupos, etc...

Como ya hemos dicho, aunque los distintos sistemas P2P basados en DHTs se comportan de la misma forma desde el punto de vista del nivel de aplicación, existen muy diversas topologías y protocolos de enrutado asociados a dichas topologías. Recientemente se han publicado estudios ([17]) acerca de cual de las distintas topologías son las más adecuadas y como éstas pueden influir en la eficiencia y en la latencia de una red P2P, ya que, aunque todas ellas garantizan enrutar un mensaje en un tiempo de orden $O(logN)$ en condiciones normales, las diferencias existentes entre unas y otras varían el comportamiento ante fallos masivos del sistema.

En la siguiente Sección realizamos un repaso a las distintas topologías que existen en redes P2P basadas en DHTs y sus implementaciones.

3 Redes P2P Estructuradas Basadas en DHTs

Pese a que el comportamiento de un sistema P2P basado en DHTs desde el punto de vista del nivel de aplicación es el mismo para cualquier implementación, la red subyacente de nodos pude seguir topologías muy diversas. En esta Sección mostramos algunas de estas topologías junto con su implementación más extendida.

3.1 Hipercubos, Content Addressable Network (CAN)

Content Addressable Network (CAN); propuesta por primera vez en [18] es un sistema P2P que provee funcionalidades de tabla hash de forma distribuida. CAN fué diseñada para ser escalable, tolerante a fallos y autorganizada. El diseño básico de su arquitectura es un espacio de coordenadas Cartesianas multidimensional sobre un toro, siendo d-dimensional el espacio lógico sobre el que se mapean las claves del sistema y los nodos que lo componen. Según [17], si el número d de dimensiones es $logN$, donde N es el número de nodos del sistema, podemos considerar que CAN sigue una topología de hipercubo $logN$-dimensional. En este espacio de coordenadas a cada nodo se le asigna una partición del espacio, de tal forma que a cada nodo le corresponde una zona única y diferenciada. En CAN, un nodo mantiene una tabla de enrutado con las direcciones IP y la zona de coordenadas virtual que le corresponden a sus vecinos en el espacio de coordenadas. Usando estas coordenadas, un nodo es capaz de enrutar un mensaje

hacia su destino usando un simple algoritmo voraz que reenvía dicho mensaje hacia aquel de entre sus vecinos que esta más cerca del destino en el sistema de coordenadas.

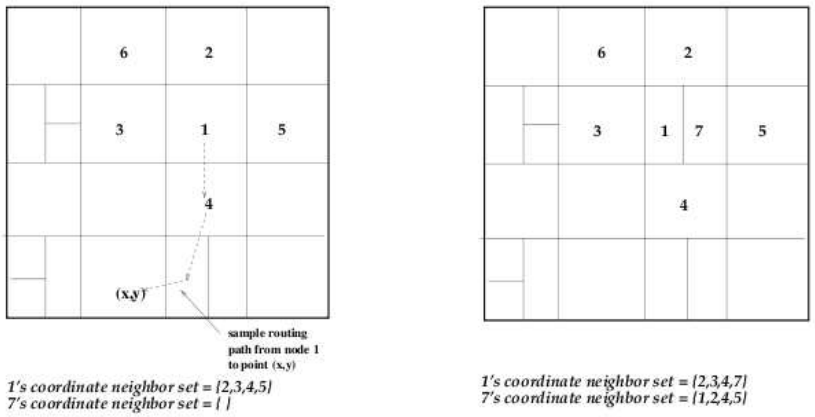

Fig. 3. Sistema CAN de 2D antes y después de que cierto nodo 7 entre al sistema. También se muestra el camino óptimo entre 1 y el punto (x, y).

Como podemos ver en la Figura 3 extraída de [18], el espacio de coordenadas es usado para almacenar pares del tipo {clave, valor}, para ello se usa una función determinista que a cada clave K posible se le asigna un punto P en el espacio de coordenadas usando una función hash. El protocolo de búsqueda garantiza que cualquier nodo puede aplicar la misma función hash y obtener el mismo punto P del espacio, de tal forma que ejecutando el protocolo de enrutado ya descrito es posible llegar al nodo responsable de dicho sector del espacio con un coste de $O(d * N^{1/d})$.

Cuando un nuevo nodo a desea acceder al sistema, debe ejecutar el algoritmo de arranque, o bootstrapping, correspondiente. En CAN existe una mecanismo de DNS que permite localizar nodos del sistema mediante un nombre DNS. Una vez que el nodo ha recuperado la IP de algún nodo del sistema, se pone en contacto con él indicándole su deseo de entrar a formar parte del sistema. Para ello a elige un punto al azar P del espacio de coordenadas y lo transmite al nodo del sistema, éste usa el protocolo de enrutado para enviar dicho mensaje al punto P de la red. Una vez llegue a su destino, el nodo q encargado de dicho punto dividirá su espacio de coordenadas en 2 quedandos él con una mitad y dándole la otra al nuevo nodo. Una vez a esté conectado a q, éste último comunica a a su tabla de vecinos con objeto de que a construya la lista de nodos vecinos de las regiones adyacentes a la suya.

En el caso de que algún nodo x abandone el sistema, existe un algoritmo de reestructuración encargado de que alguno de los vecinos del nodo x tome el control sobre la zona anteriormente controlada por x. Una vez hecho esto, el nuevo encargado de la zona informa a sus vecinos del cambio para que estos mantengan actualizada su tabla de vecinos. El número de vecinos que posee un cierto nodo depende sólo del numero d de dimensiones que posea el sistema, por ejemplo $2 * d$, y nunca del número total de nodos contenidos en el mismo.

Una forma sencilla de mejorar este sistema y de añadirle replicación sería la de mantener un sistema con r sistemas de coordenadas, cada uno de los cuales se llama *reality* o *realidad*. Cada nodo de este *multiverso* poseería unas coordenadas distintas en cada una de las realidades con una lista de vecinos totalmente distinta para cada una de las realidades. De esta forma, cada par {clave, valor} que se insertase en el sistema correspondería a un área del espacio distinto para cada una de las realidades, siendo almacenado por r nodos distintos.

3.2 Anillos

Chord Chord [19] es uno de los múltiples sistemas que usan una topología en forma de anillo. Chord usa una función de hashing consistente para asignar claves a los nodos de tal forma que el impacto de un nodo al entrar y salir del sistema sea mínimo. Este esquema descentralizado esta balanceado debido a que a cada nodo se le asigna el mismo número de claves, es por esto que cada vez que el sistema cambia, el intercambio de claves entre nodos es mínimo. Los nodos en Chord se organizan siguiendo una topología en anillo, donde cada nodo está conectado con el siguiente nodo en el anillo y con sólo otros $O(logN)$ nodos del sistema, siendo N el número de nodos del sistema.

La función de hashing que se usa para generar las claves tanto para información como para nodos es SHA-1, la cual genera claves de 160 bits. Para el caso de los identificadores de los nodos, se aplica SHA-1 a la dirección IP del nodo, garantizando asi que ningún otro nodo tendrá el mismo identificador. En cuanto a la información almacenada en el sistema, para generar la clave se aplica la función hash a la propia información. Se ha elegido SHA-1 por que es necesario que la longitud de clave generada, m, por la funcion hash sea lo suficientemente grande para garantizar que no habrá colisiones, es decir distintas claves con el mismo identificador. Este espacio de claves se mapea sobre un círculo módulo $2m$. En este círculo, cada clave k es asignada al primer nodo cuyo identificador sea mayor o igual al suyo, dicho nodo se conoce como el $sucesor(k)$ y representa el siguiente nodo en el círculo en sentido horario despues de k. Cuando un nodo n entra en el sistema, algunas de las claves anteriormente asignadas al $sucesor(n)$ son reasignadas a n. De la misma forma, cuando n deja el sistema, las claves de las que es responsable son reasignadas al $sucesor(n)$. En la Figura 4, extraída de [1], podemos ver un ejemplo del funcionamiento de un sistema Chord: para un anillo con 10 nodos, el nodo N8 intenta encontrar la clave K54, para ello se va comunicando sucesivamente con sus vecinos hasta encontrar el nodo que posee K54.

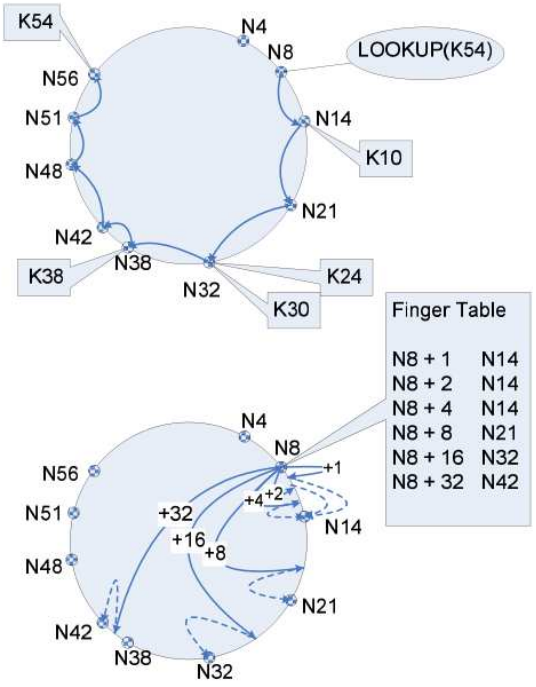

Fig. 4. Operación de búsqueda sobre un anillo Chord.

Pastry De la misma forma que Chord, Patry [20] organiza su espacio de claves y de nodos en un círculo, y de la misma forma, un identificador en Pastry indica una posición en el círculo de claves. Sin embargo, la forma de enrutar no es siguiendo la cadena de sucesores, si no reenviando a aquellos vecinos en el anillo cuya distancia al destino sea la menor, algoritmo propuesto por Plaxton en [12]. La forma de obtener dicha distancia es midiendo el tamaño del prefijo del identificador que encaja bit a bit con el identificador de la clave buscada.

3.3 Topología basada en una métrica XOR, Kademlia

Kademlia [21], implementada en los sistemas de intercambio de archivos Emule y Bitorrent, sigue la misma táctica que los anteriores de asignar a cada nodo un identificador de 160 bits contenido dentro del espacio de claves. Usa un sistema de enrutado orientado al identificador del nodo que está basado en el reenvió de mensajes al vecino que esté más próximo. La diferencia con el resto de DHTs es la métrica usada para calcular la distancia entre dos puntos del sistema. Kademlia usa una función XOR para calcular esta distancia debido a que esta función es simétrica. Kademlia se aprovecha de esta propiedad para aprender sobre las rutas en cada operación de búsqueda, ya que la función XOR asegura que todas las rutas que van hacia cierto punto del sistema terminan convergiendo.

4 Computación sobre redes P2P

Desde el surgimiento de las redes P2P se ha debatido mucho sobre la posibilidad de usar este paradigma de sistema distribuido como base para montar un sistema de computación [22]. Un sistema de computación distribuido se puede definir como un conjunto de computadores interconectados entre si por una red de comunicaciones que intentan unir sus recursos para llevar a cabo algun tipo de tarea computacional. Cada computador conectado al sistema posee sus propios recursos computacionales independientes, sin embargo, desde el punto de vista del usuario, el sistema se percibe como un único sistema. En dicho sistema un usuario accedería a los recursos de la misma forma en que accede a sus recursos locales. Sin embargo, el hecho de que el sistema esté repartido entre múltiples entidades le da unas características en cuanto a escalabilidad y soporte a fallos que muy pocos poseen.

Sistemás que cumplan estas condiciones hay muchos, desde *clústeres* hasta sistemas de *Grid Computing*, pasando por sistemas de *Desktop Grid*. Entre todos estos sistemas podemos destacar algunas soluciones como Globus en sistemas de Grid, Condor en sistemas de Clustering o Boinc en cuanto a sistemas de Desktop Grid. Todos estos sistemas, sin embargo, coinciden en un punto: todos están centralizados en algún punto. Esta centralización bien sea para facilitar tareas de administración y mantenimiento o bien sea por cuestiones corporativas le priva al sistema de algunas de sus más preciadas características en cuanto a soporte de fallos o escalabilidad ya que es precisamente este punto central el eslabón más débil de la cadena y por donde el sistema entero puede fallar.

Un sistema de computación mediante redes P2P supera esta desventaja al otorgarle a sus integrantes las labores administrativas y al darles iguales derechos ante el resto de los nodos de la red. Sin embargo se deben solventar una serie de problemas técnicos sobre los que se está trabajando en la actualidad.

Volatilidad. La gran volatilidad de un sistema P2P hace que sea necesario implementar algún sistema de checkpointing que permita parar la ejecución de un proceso en un nodo y retomarla en otro distinto. Existen diferentes soluciones y avances en sistemas de grid [23] y en sistemas de tipo OpenMosix [24]. Entre dichas soluciones podemos encontrar varios sistemas de checkpointing utilizados actualmente: CHPOX [25] usado por OpenMosix, DMTCP [26] o libckpt [27]. En [28] se evalúa el impacto de aplicar Checkpointing a la computación distribuida en redes P2P y sistemas de Grid computing. En dicho articulo se propone la realización de Checkpointings periódicos durante la ejecución de una tarea con objeto de utilizar las imágenes intermedias de la ejecución como puntos de partida para otros nodos, replicando estas subtareas como medida de backup. Otra solución propuesta en [29] consiste en predecir el tiempo que un nodo estará conectado a un sistema P2P y adaptar el scheduling de dichos sistemas a dichas predicciones maximizando así el número de ejecuciones finalizadas.

Heterogeneidad de sistemas. En un sistema de computación voluntaria como sería un sistema de computación en redes P2P se debe tener en cuenta la heterogeneidad de sistemas que se conectarán a la misma, no sólo ya en cuanto

a hardware si no también a nivel de sistema operativo o de librerías presentes en el sistema. Existen dos soluciones a priori a este problema: o bien se dividen los usuarios por grupos de iguales o bien se utiliza virtualización. Existen propuestas para usar virtualización en sistemas de tipo Boinc [30] o de tipo Grid [31] en las cuales se propone que cada tarea a ejecutar en el sistema sea una instancia de una máquina virtual. Sin embargo la perdida de rendimiento en cuanto a carga de CPU o en cuanto a ancho de banda no parece justificar esta medida. En [32] se ha estudiado el impacto que sobre una aplicación tiene ser ejecutada sobre una máquina virtual y el impacto causado sobre sistemas de Grid Computing por usar virtualización.Tal vez una solución se el uso de virtualización sólo en el caso de que sea necesario, es decir, se podrían definir una serie de plataformas estándares y que cada nodo contara con un hypervisor con dichas plataformas en hibernación de tal forma que si se da el caso de que no exista ningún nodo con el sistema operativo necesario para ejecutar una tarea que precise dicho sistema operativo, se elija un nodo libre del sistema y dicho nodo ejecute la tarea en la instancia estándar del sistema operativo requerido que tenía en hibernación. Con esto se ganaría en cuanto a ancho de banda, ya que no habría que transferir toda la máquina virtual a ser ejecutada si no sólo la aplicación. Además se ganaría en tiempo de arranque de la tarea ya que es mucho más rápido despertar una máquina virtual hibernada que arrancarla desde cero. Sobre este aspecto, el sistema de Grid Computing *Entropia* [33] propone el uso de una tecnología de máquinas virtuales propia para distribuir tareas en un sistema distribuido, utilizando para ello máquinas virtuales estándar y distribuyendo solamente los ejecutables de la tarea a ejecutar. Otro punto importante para solucionar el problema de la heterogeneidad de sistemas, es crear sistemas que permitan definir una tarea junto con todos sus parámetros. A este referente se ha propuesto el uso de *GenWrapper* [34], un entorno de shell basado en POSIX permitiendo definir tareas a lanzar en sistemas de Grid Computing y Desktop Computing.

Seguridad. Una de las cuestiones más importantes es cómo asegurar que lo que un nodo ha devuelto como resultado de la ejecución de una tarea es correcto. Ante este problema se pueden instaurar sistemas de créditos para poder puntuar el comportamiento de un nodo en el sistema, siendo de esta forma posible desterrar nodos no deseados con el consenso de la mayoría. En [35] se propone insertar en las tareas una serie de tareas "quiz" fáciles de verificar para comprobar el nivel de confianza que se puede poner en los resultados provenientes de un nodo determinado. En la literatura se menciona sistemas basados en la reputación como la solución a este problema. Diversos métodos se han propuesto para evaluar la reputación de los nodos participantes en un sistema P2P [36–38].

4.1 Sistemás de computación P2P existentes

WaveGrid Este sistema propuesto en [39–41] consiste en un overlay DHT, soporta varios, entre ellos CAN, Chord y Pastry. Los nodos se dividen en husos

horarios de tal forma que en cada momento de tiempo, sólo se están usando aquellos nodos que se encuentren en una zona donde sea de noche y los nodos se encuentren inactivos. En el momento en que un huso horario deja de estar en una zona nocturna, se procede a realizar checkpointing y migración de la tarea hacia otra zona horaria que se encuentre en horario nocturno. De esta forma una tarea que requiera un tiempo considerable puede ir migrando alrededor del globo usando siempre ordenadores inactivos hasta que termine.

La forma de separar el overlay en diversas zonas horarias dependerá del overlay elegido. Para el caso de CAN se puede usar una de las dimensiones para discernir entre husos horarios, de tal forma que a los nodos situados en las coordenadas {1,...} les corresponderán el primer huso, a los situados en las coordenadas {2,...} el segundo y así sucesivamente.

Cuando un nodo desea lanzar un trabajo, mediante un proceso de scheduling elige de entre los nodos disponibles, aquellos que se ajusten a las condiciones que él imponga para el trabajo, y de entre esos elige el nodo más apropiado según sus criterios. Para buscar dichos nodos libres puede seguir dos procedimientos: o bien elige un punto al azar en el overlay y le pregunta al nodo encargado de dicho nodo sobre si está libre o no, y si no lo está elige otro punto aleatoriamente; o bien ejecuta una técnica de expansión del anillo de búsqueda, es decir, pregunta a sus vecinos sobre su disponibilidad, y si ninguno lo está, va preguntando sucesivamente a los vecinos de sus vecinos, hasta llegar a un límite determinado. Estos sistemas tienen el problema de que el sistema esté demasiado sobrecargado y que nunca llegue a encontrar un nodo libre, y de todas formas, el hallazgo de un nodo libre no implica que vaya a estarlo hasta que él decida ocuparlo, ni mucho menos mantiene ningún orden entre las tareas que se lanzan en el sistema.

P2P task scheduling in computation grid Propuesto en en [42] es un intento por descentralizar parte de la infraestructura de un sistema de Grid Computing como Globus. Mediante un nuevo módulo llamado Peer in Grid Scheduler (PGS) permite que los nodos que conforman el sistema sean los encargados de decidir como ejecutar las tareas que deseen lanzar. Cada nodo es responsable del scheduling de sus tareas y para ello usa el Grid Peer Information Service (GPIS), una especie de paginas amarillas centralizadas del sistema de grid, con objeto de encontrar nodos disponibles y en caso de encontrarlos lanzar las tareas a ejecutar en dicho nodo. Sin embargo, no existe ningún tipo de acuerdo en común entre los nodos para establecer un orden entre dichas tareas y, de la misma forma que con el WaveGrid, no existe una forma de lanzar una tarea en un sistema que esté totalmente ocupado, a menos que se use el GPIS para ello debido a su centralismo, pero no parece el caso.

CompuP2P CompuP2P [43] es un sistema P2P basado en DHT, actualmente implementado con Chord que divide el conjunto de los nodos del sistema entre compradores y vendedores de recursos. Asi mismo, en cada sistema CompuP2P se crean una serie de mercados con objeto de cubrir todas las posibles demandas de los nodos del sistema en cuanto a CPU, uso de disco, sistema operativo, etc.

En cada uno de estos mercados se elige a un nodo propietario del mercado que es el encargado de realizar las búsquedas de recursos disponibles cuando se lanza una tarea. Cuando un nodo desea lanzar una tarea con ciertos requerimientos, se dirige al propietario del mercado que le interese haciéndole llegar sus necesidades. El propietario busca entre los recursos que controla aquel que esté libre y que cumpla con los requerimientos del nodo cliente. Una vez el nodo cliente conoce los datos del nodo dispuesto a ejecutar su tarea, se pone en contacto con él para ultimar los detalles de la operación.

CoDiP2P Sistema desarrollado por la Universidad de Lleida, CoDiP2P [44] es un sistema basado en una red P2P estructurada en forma de árbol pero que no usa DHT. Basa su funcionamiento en mapear en dicha estructura en árbol los nodos del sistema diferenciando entre nodos maestros y nodos esclavos por regiones. Cuando un nodo desea lanzar una tarea, se pone en contacto con el nodo maestro de su zona, el cual busca los nodos libres de la zona y decide la mejor forma de ejecutar la tarea. En el caso de que no exista ningún nodo libre en la zona, se comunicará con su nodo padre en el arbol el cual actua como nodo maestro y será el encargado de buscar los recursos disponibles en todos sus nodos hijos.

Claramente este sistema se separa de las bases de los sistemas P2P debido a la centralización que impone a sus nodos. Aunque en este caso si se asegura la ejecución de todas las tareas que se lancen en el sistema, esto se garantiza mediante la utilización de servidores centralizados.

Super-Peer Approach for Public Scientific Computation Este sistema propuesto en [45] intenta unificar los conceptos de Grid Computing y P2P Computing. Para ello propone una aproximación ya usada por Napster o Gnutella: redes P2P de "Super-Peers". Dichos Super-Peers están al cargo de una serie de nodos Workers conectados mediante el modelo Cliente-Servidor y encargados de ejecutar las tareas que el Super-Peer considere apropiadas dada su carga y características. Por otro lado todos los Super-Peers del sistema conforman un overlay P2P mediante el cual se informan entre ellos de los trabajos y de los nodos disponibles con lo que cuentan.

Cuando un nodo Worker considera que está libre, solicita una tarea a su Super-Peer informándole de sus características y disponibilidad. El Super-Peer realizará una búsqueda en su caché local o en el overlay de Super-Peers hasta encontrar un trabajo que se adapte a dicho nodo. Una vez asignado una tarea, el Worker iniciará una comunicación con los nodos cacheadores de datos solicitando los datos necesarios para ejecutar la tarea.

4.2 High Throughput Computing sobre Redes P2P

Como se ha podido ver en apartados anteriores, uno de los mayores problemas de los sistemas P2P para computación existentes es que en ninguno de ellos se

garantiza la ejecución de una tarea, ya que si el sistema se encuentra sobrecargado, no existe ningún mecanismo que asegure que la tarea se terminará por ejecutar en algún momento del tiempo de vida del sistema. En la mayoría de estos sistemas, si el nodo que desea lanzar una tarea es incapaz de encontrar un recurso disponible, nunca se lanzarán las tareas de dicho nodo. Es más, al no existir ningún tipo de orden entre las tareas que se ejecutan en el sistema, nadie garantiza que todas las tareas lanzadas en el sistema se ejecuten tarde o temprano, pudiéndose llegar a producir un situación conocida como "starvation".

Es por esto que en los sistemas de computación distribuida habitualmente existen sistemas de colas encargados de mantener un cierto orden temporal en las tareas que se ejecutan en el sistema de tal forma que todas las tareas se ejecuten tarde o temprano. Por supuesto existen excepciones en este tipo de sistemas. Usando una técnica llamada "backfilling" es posible adelantar tareas de tal forma que si no existen recursos disponibles para ejecutar la primera tarea de la cola pero si que existen para ejecutar alguna de las siguientes, el sistema le dé más prioridad a esta segunda tarea adelantando su ejecución con objeto de tener el sistema funcionando el máximo tiempo posible. La implementación sobre un sistema de computación P2P de un sistema de colas permitiría la ejecución en orden de tareas y diversas técnicas de scheduling mediante backfilling.

Según [15] es posible construir un sistema de comunicación mediante multicast y anycast a grupos en DHTs, con lo cual sería posible construir y mantener grupos con intereses comunes usando simplemente las herramientas de la propia DHT. Además, si añadimos a un protocolo de DHT la replicación de claves por el simple método de insertar en el sistema copias de una cierta clave k refiriéndose a ellas como $\{k:1\}..\{k:n\}$ para un cierto número de réplicas n, entonces sería posible construir un sistema de almacenaje distribuido en el propio overlay DHT. En la literatura existen múltiples proposiciones [46–52] para convertir un overlay P2P basado en DHT en un sistema de almacenaje que asegure la permanencia y la consistencia de los objetos almacenados usando para ello multiples réplicas de dichos objetos y usando distintos algoritmos para mantener la coherencia entre dichas réplicas.

En [15, 53, 54] se proponen varios algoritmos para acceder mediante exclusión mutua a un sistema distribuido. El principio fundamental es el concepto de torneos por rondas con backoff: en cada ronda, los nodos deseosos de entrar en la sección crítica (SC) deben comunicárselo a cada uno de los n nodos que componen dicha SC. Si consigue hacerse con el control de al menos $n/2+1$ nodos de la sección crítica, entonces se le declara vencedor y puede entrar en la sección crítica. Si no, entonces esperará durante un cierto tiempo que puede ir aumentando de forma exponencial con cada ronda que falle en ganar (backoff). Usando estos algoritmos es posible construir un sistema que almacene objetos manteniendo la coherencia entre las réplicas de dichos objetos ya que la modificación de cualquier objeto se realiza dentro de una sección crítica y por lo tanto cualquier modificación en una réplica se propaga a las demás.

Sobre este sistema de almacenamiento con réplicas y comunicación a grupos es posible crear grupos en la DHT que mantengan un objeto cola de procesos

de forma distribuida y mediante la cual se mantenga el orden de ejecución de las tareas que se lancen en el sistema. Los nodos libres del sistema que deseen ejecutar alguna tarea se harían dueños del objeto usando exclusión mutua para extraer de la misma la primera tarea dependiendo de la política de backfilling usada y ejecutaría la tarea.

Además este sistema permitiría que un cierto nodo depositara la información de la tarea que desease lanzar en la cola distribuida junto con los datos necesarios para lanzar la tarea (ejecutables, librerías, datos de configuración...) en el DHT de tal forma que podría desconectarse del sistema y su tarea sería eventualmente lanzada sin su colaboración. Cuando este nodo volviese a entrar al sistema encontraría los resultados de la ejecución que solicitó almacenados en el sistema DHT, pudiendo así recuperarlos dentro de un tiempo razonable.

5 Herramientas para el estudio de las redes P2P

Existen una serie de herramientas que permiten el estudio, la validación y la implementación de sistemas basados en redes P2P asi como de nuevas redes P2P de forma relativamente rápida. Por un lado tenemos los simuladores que nos permiten testear nuestras propuestas y por otro las librerías que implementan ciertas redes ya existentes.

5.1 Simuladores

OverSim Descrito en [55], OverSim es un simulador de eventos basado en OMNeT++ que permite simular cualquier red P2P, estructuradas o no, y que implementa varios protocolos P2P, entre ellos Chord y Kademlia. Es fácilmente ampliable mediante módulos programados en C++ y permite la simulación de redes P2P con diversos grados de detalle en función de qué módulo se use para simular la red subyacente.

P2PSim Propuesto en [56], es un simulador de eventos a nivel de paquete que puede simular sólo overlays estructurados. Posee implementaciones de Chord, Accordion, Koorde, Kelips, Tapestry y Kademlia. La API en C++ está pobremente documentada pero existen clases ejemplo para facilitar la implementación de nuevos protocolos extendiéndolas. Actualmente su desarrollo está abandonado.

PeerSim Simulador de eventos accesible en [57]. Está desarrollado en Java y publicado bajo la licencia GPL. Permite la simulación de redes P2P estructuradas y no estructuradas. Es ampliable mediante módulos e incluye implementaciones de Chord y Pastry entre otros.

PlanetSim [58] Simulador por eventos desarrollado en Java. Permite simular redes estructuradas y no estructuradas. Además incluye la implementación de Chord y Symphony así como también implementa la Common API. Por desgracia no permite la obtención de estadísticas, con lo que su uso es más bien anecdótico.

PeerSE [59] Simulador desarrollado en Java. Permite la simulación tanto del overlay como del nivel físico.

5.2 Librerías

Existen diversas implementación disponibles de las redes P2P comentadas en este informe, entre ellas podemos encontrar:

Chord `http://pdos.csail.mit.edu/chord/`.

Pastry `http://freepastry.org/`.

Kademlia Tanto los clientes de la red Emule como los de la red Bitorrent implementan esta red P2P, sin embargo existen una serie de implementaciones en forma de librería: `http://kadc.sourceforge.net/` (en C), `http://khashmir.sourceforge.net/` y `http://www.heim-d.uni-sb.de/~heikowu/SharkyPy/` (en Python) y `http://www.thomÃąs.ambus.dk/plan-x/routing/` (en Java).

6 Plan de trabajo propuesto

1. Seleccionar el overlay DHT más apto para un sistema de HTC sobre P2P. ¿Posible articulo comparativo?
2. Dotar a dicho overlay de capacidad para almacenar permanentemente objetos en la DHT. Para ello habrá que elegir el algoritmo de replicación más apto o proponer uno.
3. Dotar al sistema de capacidad para realizar operaciones en exclusión mutua.
4. Sobre dicho sistema, construir un sistema de colas distribuido.

7 Conclusiones

En el presente informe se ha dado una visión general del estado del arte en cuanto a redes Peer-to-Peer y especialmente en lo que se refiere a sistemas de computación distribuida sobre dichos sistemas. Se han mostrado sus características y sus defectos. También se han aportado una serie de ideas para mejorar estos sistemas. Por último se han mostrado una serie de herramientas en forma de simuladores y librerías que pueden ser útiles para el estudio de las redes P2P.

References

1. Lua, E.K., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys and Tutorials **7**(1-4) (2005) 72–93
2. Perkins, C.E., Bhagwat, P.: Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In: SIGCOMM. (1994) 234–244
3. Hsiao-Kuang, C.C.C., chuan Chiang, C., kuang Wu, H., Liu, W., Gerla, M.: Routing in clustered multihop, mobile wireless networks with fading channel (1997)
4. Belding-Royer, E.M., Perkins, C.E.: Multicast operation of the ad-hoc on-demand distance vector routing protocol. In: MOBICOM. (1999) 207–218
5. Park, V.D., Corson, M.S.: A highly adaptive distributed routing algorithm for mobile wireless networks. In: INFOCOM. (1997) 1405–1413
6. Ho, Y.K., Liu, R.S.: A novel routing protocol for supporting qos for ad hoc mobile wireless networks. Wirel. Pers. Commun. **22**(3) (2002) 359–385
7. Royer, E.M., Toh, C.K.: A review of current routing protocols for ad hoc mobile wireless networks (1999)
8. The Napster web site. `http://www.napster.com.`
9. The Gnutella web site. `http://gnutella.wego.com`
10. The Gnutella Protocol Specification. `http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html`
11. Gnucleus, the gnutella web caching system. `http://www.gnucleus.com/gwebcache/`
12. Plaxton, C.G., Rajaraman, R., Richa, A.W.: Accessing nearby copies of replicated objects in a distributed environment. Theory Comput. Syst. **32**(3) (1999) 241–280
13. Godfrey, B., Lakshminarayanan, K., Surana, S., Karp, R.M., Stoica, I.: Load balancing in dynamic structured p2p systems. In: INFOCOM. (2004)
14. Godfrey, B., Shenker, S., Stoica, I.: Minimizing churn in distributed systems. In Rizzo, L., Anderson, T.E., McKeown, N., eds.: SIGCOMM, ACM (2006) 147–158
15. Dabek, F., Zhao, B.Y., Druschel, P., Kubiatowicz, J., Stoica, I.: Towards a common api for structured peer-to-peer overlays. [60] 33–44
16. Rhea, S.C., Godfrey, B., Karp, B., Kubiatowicz, J., Ratnasamy, S., Shenker, S., Stoica, I., Yu, H.: Opendht: a public dht service and its uses. In Guérin, R., Govindan, R., Minshall, G., eds.: SIGCOMM, ACM (2005) 73–84
17. Gummadi, P.K., Gummadi, R., Gribble, S.D., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of dht routing geometry on resilience and proximity. In Feldmann, A., Zitterbart, M., Crowcroft, J., Wetherall, D., eds.: SIGCOMM, ACM (2003) 381–394
18. Ratnasamy, S., Francis, P., Handley, M., Karp, R.M., Shenker, S.: A scalable content-addressable network. In: SIGCOMM. (2001) 161–172
19. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Netw. **11**(1) (2003) 17–32
20. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In Guerraoui, R., ed.: Middleware. Volume 2218 of Lecture Notes in Computer Science., Springer (2001) 329–350
21. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In Druschel, P., Kaashoek, M.F., Rowstron, A.I.T., eds.: IPTPS. Volume 2429 of Lecture Notes in Computer Science., Springer (2002) 53–65

22. Foster, I.T., Iamnitchi, A.: On death, taxes, and the convergence of peer-to-peer and grid computing. [60] 118–128
23. Chtepen, M., Claeys, F.H.A., Dhoedt, B., Turck, F.D., Demeester, P., Vanrolleghem, P.A.: Adaptive task checkpointing and replication: Toward efficient fault-tolerant grids. IEEE Trans. Parallel Distrib. Syst. **20**(2) (2009) 180–190
24. chuen Ho, R.S., Wang, C.L., Lau, F.C.: Lightweight process migration and memory prefetching in openmosix. In: IPDPS, IEEE (2008) 1–12
25. the chpox web site. `http://freshmeat.net/projects/chpox/`
26. The DMTCP web site. `http://dmtcp.sourceforge.net/`
27. The libckpt web site. `http://www.cs.utk.edu/~plank/plank/www/libckpt.html`
28. Gil, J.M., Song, U.S., Yu, H.C.: Performance evaluation of scheduling mechanism with checkpoint sharing and task duplication in p2p-based pc grid computing. [61] 459–470
29. Zhang, H., Jin, H., Zhang, Q.: Scheduling strategy of p2p based high performance computing platform base on session time prediction. [61] 364–375
30. Gonzalez, D.L., de Vega, F.F., Trujillo, L., Olague, G., Araujo, L., Castillo, P.A., Guervós, J.J.M., Sharman, K.: Increasing gp computing power for free via desktop grid computing and virtualization. [62] 419–423
31. Pradheep, S.S., Santhanam, S., Elango, P., Arpaci-dusseau, A., Livny, M.: Deploying virtual machines as sandboxes for the grid. In: In Second Workshop on Real, Large Distributed Systems (WORLDS 2005). (2005) 712
32. Domingues, P., Araujo, F., Silva, L.: Evaluating the Performance and Intrusiveness of Virtual Machines for Desktop Grid Computing. In: 2009 IEEE INTERNATIONAL SYMPOSIUM ON PARALLEL & DISTRIBUTED PROCESSING, VOLS 1-5. International Parallel and Distributed Processing Symposium (IPDPS), 345 E 47TH ST, NEW YORK, NY 10017 USA, IEEE, IEEE (2009) 2349–2356 23rd IEEE International Parallel and Distributed Processing Symposium, Rome, ITALY, MAY 23-29, 2009.
33. Chien, A.A., Calder, B., Elbert, S., Bhatia, K.: Entropia: architecture and performance of an enterprise desktop grid system. J. Parallel Distrib. Comput. **63**(5) (2003) 597–610
34. Marosi, C.A., Balaton, Z., Kacsuk, P.: Genwrapper: A generic wrapper for running legacy applications on desktop grids. In: IPDPS, IEEE (2009) 1–6
35. Zhao, S., Lo, V.M., GauthierDickey, C.: Result verification and trust-based scheduling in peer-to-peer grids. In Caronni, G., Weiler, N., Waldvogel, M., Shahmehri, N., eds.: Peer-to-Peer Computing, IEEE Computer Society (2005) 31–38
36. Xie, Z., Bi, J.: PeerStrategy: A local strategy for peers to evaluate their neighbors. In: GCC 2008: SEVENTH INTERNATIONAL CONFERENCE ON GRID AND COOPERATIVE COMPUTING, PROCEEDINGS, 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA, Chinese Acad Sci, Shenzhen Inst Adv Technol; Dawning Informat Ind Co Ltd; Chinese Natl Grid; European & Chinese Cooperat Grid; Chinese Acad Sci Shenzhen Acad Consultat Ctr; Chinese Acad Engn Shenzhen Acad Consultat Ctr, IEEE COMPUTER SOC (2008) 386–391 7th International Conference on Grid and Cooperative Computing, Shenzhen, PEOPLES R CHINA, OCT 24-26, 2008.
37. Fedotova, N., Orzetti, G., Veltri, L., Zaccagnini, A.: Byzantine Agreement for Reputation Management in DHT-based Peer-to-Peer Networks. In: 2008 INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, VOLS 1 AND 2, 345 E 47TH ST, NEW YORK, NY 10017 USA, IEEE (2008) 164–169 International Conference on Telecommunications, St Petersburg, RUSSIA, JUN 16-19, 2008.

38. Heien, E.M., Anderson, D.P., Hagihara, K.: Computing Low Latency Batches with Unreliable Workers in Volunteer Computing Environments. JOURNAL OF GRID COMPUTING **7**(4, Sp. Iss. SI) (DEC 2009) 501–518
39. Lo, V.M., Zappala, D., Zhou, D., Liu, Y., Zhao, S.: Cluster computing on the fly: P2p scheduling of idle cycles in the internet. [63] 227–236
40. Zhou, D., Lo, V.M.: Wave scheduler: Scheduling for faster turnaround time in peer-based desktop grid systems. In Feitelson, D.G., Frachtenberg, E., Rudolph, L., Schwiegelshohn, U., eds.: JSSPP. Volume 3834 of Lecture Notes in Computer Science., Springer (2005) 194–218
41. Zhou, D., Lo, V.M.: Wavegrid: a scalable fast-turnaround heterogeneous peer-based desktop grid system. In: IPDPS, IEEE (2006)
42. Cao, J., Kwong, O.M.K., Wang, X., Cai, W.: A peer-to-peer approach to task scheduling in computation grid. In Li, M., Sun, X.H., Deng, Q., Ni, J., eds.: GCC (1). Volume 3032 of Lecture Notes in Computer Science., Springer (2003) 316–323
43. Gupta, R., Sekhri, V., Somani, A.K.: Compup2p: An architecture for internet computing using peer-to-peer networks. IEEE Trans. Parallel Distrib. Syst. **17**(11) (2006) 1306–1320
44. Martínez, D.C., Torrento, J.R., Vilardell, I.B., de Sola, F.G., Solsona, F.: A new reliable proposal to manage dynamic resources in a computing p2p system. [62] 323–329
45. Mastroianni, C., Cozza, P., Talia, D., Kelley, I., Taylor, I.: A scalable super-peer approach for public scientific computation. Future Gener. Comput. Syst. **25**(3) (2009) 213–223
46. Knezevic, P., Wombacher, A., Risse, T.: Enabling high data availability in a dht. In: DEXA Workshops, IEEE Computer Society (2005) 363–367
47. Chun, B.G., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, M.F., Kubiatowicz, J., Morris, R.: Efficient replica maintenance for distributed storage systems. In: NSDI, USENIX (2006)
48. Dabek, F., Kaashoek, M.F., Karger, D.R., Morris, R., Stoica, I.: Wide-area cooperative storage with cfs. In: SOSP. (2001) 202–215
49. Kim, K., Park, D.: Reducing replication overhead for data durability in dht based p2p system. IEICE Transactions **90-D**(9) (2007) 1452–1455
50. Zhao, J., Yu, H., Zhang, K., Zheng, W., Wu, J., Hu, J.: Achieving reliability through replication in a wide-area network dht storage system. In: ICPP, IEEE Computer Society (2007) 29
51. Bessa, S., Correia, M.E., Brandao, P.: Storage and retrieval on P2P networks: A DHT based protocol. In: 2007 IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, VOLS 1-3, 345 E 47TH ST, NEW YORK, NY 10017 USA, IEEE; IEEE Commun Soc; IEEE Comp Soc; Inst Telecommun; Univ Aveiro; Inovacao; AT&T, IEEE (2007) 829–835 IEEE Symposium on Computers and Communications, Santiago, PORTUGAL, JUL 01-04, 2007.
52. Kiw, K., Park, D.: Reducing replication overhead for data durability in DHT based P2P system. IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS **E90D**(9) (SEP 2007) 1452–1455
53. Lin, S., Lian, Q., Chen, M., Zhang, Z.: A practical distributed mutual exclusion protocol in dynamic peer-to-peer systems. [63] 11–21
54. Chockler, G., Malkhi, D., Reiter, M.K.: Backoff protocols for distributed mutual exclusion and ordering. In: ICDCS. (2001) 11–20
55. Baumgart, I., Heep, B., Krause, S.: OverSim: A Flexible Overlay Network Simulation Framework. In: Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA. (May 2007)

56. Barbosa, J., Hahn, R., Bonatto, D., Cecin, F., Geyer, C.: Evaluation of a large-scale ubiquitous system model through peer-to-peer protocol simulation. In: DS-RT '07: Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications, Washington, DC, USA, IEEE Computer Society (2007) 175–181
57. The PeerSim website. `http://peersim.sourceforge.net/`
58. López, P.G., Pairot, C., Mondéjar, R., Ahulló, J.P., Tejedor, H., Rallo, R.: Planetsim: A new overlay network simulation framework. In Gschwind, T., Mascolo, C., eds.: SEM. Volume 3437 of Lecture Notes in Computer Science., Springer (2004) 123–136
59. Bischofs, L., Hasselbring, W.: Analyzing and implementing peer-to-peer systems with the peerse experiment environment. [62] 311–315
60. Kaashoek, M.F., Stoica, I., eds.: Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers. In Kaashoek, M.F., Stoica, I., eds.: IPTPS. Volume 2735 of Lecture Notes in Computer Science., Springer (2003)
61. Abdennadher, N., Petcu, D., eds.: Advances in Grid and Pervasive Computing, 4th International Conference, GPC 2009, Geneva, Switzerland, May 4-8, 2009. Proceedings. In Abdennadher, N., Petcu, D., eds.: GPC. Volume 5529 of Lecture Notes in Computer Science., Springer (2009)
62. Baz, D.E., Spies, F., Gross, T., eds.: Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2009, Weimar, Germany, 18-20 Febuary 2009. In Baz, D.E., Spies, F., Gross, T., eds.: PDP, IEEE Computer Society (2009)
63. Voelker, G.M., Shenker, S., eds.: Peer-to-Peer Systems III, Third International Workshop, IPTPS 2004, La Jolla, CA, USA, February 26-27, 2004, Revised Selected Papers. In Voelker, G.M., Shenker, S., eds.: IPTPS. Volume 3279 of Lecture Notes in Computer Science., Springer (2005)