Universidad Politécnica de Cartagena
E.T.S. de Ingeniería de Telecomunicación

Espacio-Tele□o n^{0}1 (2010)
Revista de la ETSIT-UPCT

Simulación de una aplicación P2P empleando Oversim

Antonio Manuel Martínez Rojo y Juan Carlos Sánchez Aarnoutse

Departamento de Tecnologías de la Información y las Comunicaciones.
Universidad Politécnica de Cartagena.

antoniom.martinez@upct.es; juanc.sanchez@upct.es

Resumen

Oversim es un entorno de simulación de redes *overlay* de código abierto basado en el simulador de eventos discretos OMNET++. Entre las ventajas principales de OverSim se puede destacar que incluye los principales protocolos P2P (*Peer-to-Peer*) tanto estructurados como no estructurados, así como mixtos. Además permite simular todas las capas de red subyacentes (de la capa MAC en adelante) gracias al entorno INET. Por estas y otras razones, Oversim solventa varios de los inconvenientes de los simuladores P2P existentes y es considerado una de las herramientas P2P más potentes. Este artículo describe a muy bajo nivel la simulación de una aplicación P2P particular empleando OverSim. Además, contiene información sobre cómo utilizarlo en simulaciones P2P generales. El objetivo perseguido es proporcionar una introducción a esta potente herramienta para quienes trabajen en el área de P2P.

Proyecto/Grupo de investigación: Grupo de Ingeniería Telemática. Entidad financiadora: Plan Nacional I+D+i Contribución a los nuevos paradigmas y tecnologías de red para las comunicaciones del mañana (CON-PARTE-1). Código: TEC2007-67966-C03-01/TCM.

Líneas de investigación: *Telemática; Redes Overlay; Aplicaciones P2P; Anonimato; VoD.*

1. Introducción

Es bien sabido que Internet es una red de redes compuesta por una cantidad enorme de equipos interconectados entre sí. En este tipo de entornos, probar un nuevo protocolo o una nueva aplicación en desarrollo conlleva un coste asociado demasiado elevado en la mayoría de los casos. Por tanto, es altamente recomendable simular previamente los nuevos algoritmos y especificaciones.

Este hecho se amplifica si el objeto de estudio es una red o una aplicación *Peer-to-Peer* (P2P) [1] debido principalmente a dos causas. Por un lado la escalabilidad: se trata de entornos con un número masivo de nodos (1,000,000 o más) que requieren operar en entornos a gran escala, esto es, en Internet. Por otro lado, los nodos P2P suelen ser ordenadores domésticos, por tanto, están sujetos a conexiones y desconexiones impredecibles. Este comportamiento ha de ser tenido en cuenta en el diseño de las nuevas aplicaciones para que puedan adaptarse apropiadamente. Tras validar el nuevo algoritmo P2P (bajo estudio) mediante simulación, los desarrolladores tienen la posibilidad de probar una instancia real de la aplicación P2P en un entorno emulado. Un emulador de red es un programa ejecutándose en un conjunto de computadores (al menos una) que aparenta ser una red. La Figura 1 resume gráficamente este planteamiento.

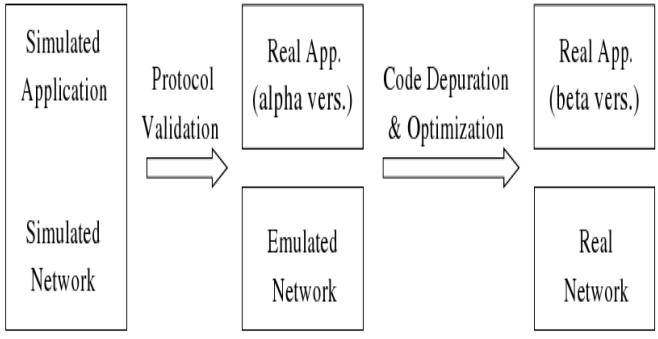

Figura 1: Procedimiento para desarrollar una aplicación de red libre de errores.

Este trabajo contiene una descripción a muy bajo nivel de cómo realizar la simulación de una aplicación P2P particular empleando el entorno de simulación OverSim [2]. Asimismo, contiene información acerca de cómo utilizarlo para aplicaciones P2P generales. OverSim está basado en el simulador por eventos discretos OMNET++, y está considerado como una de las herramientas más potentes para la comunidad investigadora.

2. Entornos de simulación para redes P2P

En esta sección se describen brevemente los entornos de simulación P2P de código abierto más relevantes (según la opinión de sus respectivos autores).

P2PSim [3] es un simulador de eventos discretos para redes P2P estructuradas escrito en C++. Cuenta con varios modelos diferentes de la red subyacente (*underlay network*). Sin embargo, apenas está documentado y por tanto resulta difícil de ampliar. Además, tiene un conjunto limitado de estadísticas.

PeerSim [4] está escrito en Java y utiliza técnicas de consultas cíclicas o de eventos discretos para simular redes P2P no estructuradas. Los componentes pueden ser implementados para recopilar datos estadísticos y puede simular hasta 1,000,000 de nodos. Sin embargo, la red TCP/IP subyacente no está modelada.

PlanetSim [5] es un simulador de eventos discretos escrito en Java. Posee un tutorial detallado, está rigurosamente documentado y una escalabilidad de 100,000 nodos. Sin embargo, el simulado de la red TCP/IP subyacente es limitado.

OverSim [6] es un entorno de simulación basado en el simulador de eventos discretos OMNeT++. Puede simular hasta 100,000 nodos y su diseño está rigurosamente documentado. Implementa tres modelos de red subyacente: (1) *Simple model*, los paquetes de datos son enviados directamente desde un nodo *overlay* hasta otro empleando una tabla de encaminamiento global. (2) *INET underlay* incluye modelos de simulación para todas las capas de red. (3) *SingleHost underlay* cada instancia de OverSim emula un único nodo, que puede ser conectado a otras instancias sobre un red real existente; esto es, los nodos *overlay* son simulados pero las capas MAC, IP, TCP/UDP trabajan en un escenario real.

En [7] puede encontrarse un estudio más exhaustivo sobre entornos de simulación P2P. De entre todas las herramientas disponibles se ha seleccionado OverSim debido a su flexibilidad: su diseño modular y la utilización de una API común [8] lo hace idóneo para desarrollar nuevos protocolos.

3. OMNeT++ y OverSim

OMNeT++ es un conocido motor de simulación de código abierto y uno de los entornos de simulación más utilizados por la comunidad investigadora. Está bien documentado y ofrece diversos modelos de simulación de red. El desarrollo de una simulación con OMNeT++ conlleva los siguientes pasos: Descripción de la estructura del sistema utilizando el lenguaje NED;

implementación de los módulos simples en C++; compilación; y configuración de la simulación.

OverSim es una herramienta de simulación *overlay* construida sobre OMNeT++. La estructura en capas de este simulador se describe en [9]. OverSim incluye todos los servicios de las redes *underlay* y *overlay* (y sus primitivas), por tanto, los usuarios sólo necesitan escribir el código de la aplicación *overlay* en la capa de aplicación. La comunicación entre la capa *overlay* y la capa de aplicación emplea la API descrita en [8]. El conjunto de funciones definidas en esta API es el mismo independientemente de la red *overlay* seleccionada. Por lo tanto, la misma aplicación puede traducirse fácilmente de un sistema *overlay* a otro con un esfuerzo mínimo. Cabe mencionar que la capa de aplicación se encuentra subdividida en dos subcapas diferentes (*Tier 1* y *Tier 2*). Una aplicación simple puede programarse en la subcapa *Tier 1*. Sin embargo, aplicaciones más complejas deben ser programadas en la subcapa *Tier 2*, sobre todo si éstas utilizan servicios proporcionados por la aplicación implementada en la subcapa *Tier 1*.

Para construir una simulación con OverSim es necesario implementar los módulos en C++ en la capa de aplicación, así como los mensajes que serán intercambiados entre ellos.

4. Redes P2P: operaciones y clasificación

En términos generales, en una aplicación P2P las operaciones más comunes son las siguientes: *publicación*: el cliente publica su lista de archivos compartidos; *búsqueda*: el cliente busca los contenidos que desea descargar; *obtención de nodos fuente*: el cliente obtiene una lista de otros clientes que poseen archivos que éste desea descargar (estos clientes se denominan "fuentes"); y *descarga*: un cliente se conecta a otro (una fuente) con el fin de descargar un archivo.

En la bibliografía se pueden distinguir tres tipos de redes P2P en función de los servicios de *búsqueda* implementados: modelo de directorio centralizado, modelo no estructurado y modelo estructurado.

El modelo de directorio centralizado está basado en un servidor de directorio que publica información sobre los contenidos ofrecidos para compartición. El directorio almacena meta-información para búsquedas. Su mayor desventaja reside en problema del punto de fallo único.

Las redes P2P no estructuradas emplean mecanismos de inundación (*flooding*) para solicitar datos. Cada petición de un *peer* es transmitida mediante *flooding* directamente hacia los *peers* a los que está conectado, y éstos a su vez reenvían dicha petición a sus *peers* directamente conectados hasta que la petición sea respondida o se alcance el número máximo de pasos de *flooding*.

Las redes P2P estructuradas están diseñadas para aplicaciones que se ejecuten sobre redes bien organizadas. El proceso de búsqueda es completado en $\mathcal{O}(\log N)$ saltos. Estos sistemas normalmente aseguran búsquedas rápidas y eficientes, pero no soportan consultas difusas (la consulta debe contener el nombre exacto del contenido buscado).

Combinando las ventajas del modelo de directorio centralizado y el sistema P2P no estructurado, muchos sistemas P2P emplean una arquitectura híbrida con dos tipos de *peers*: el *peer* común al que se denomina "*peer*", y el "*Super-Peer*" (*SP*). Todo *peer* debería conectarse a un *SP*. El *SP* actúa como un servidor centralizado en un sistema de modelo de directorio central. Los *SP*s se conectan entre sí del mismo modo que en un sistema P2P no estructurado.

5. API para la simulación P2P

En esta sección se describen las operaciones para interactuar con las aplicaciones P2P. Son tomadas en consideración únicamente las redes P2P estructuradas y redes basadas en *Super-Peer* debido a que son las más extendidas hoy en día.

5.1. Red P2P estructurada

Como se ha mencionado anteriormente, en OverSim la comunicación con la capa overlay emplea la API descrita en [8]. Ésta de?ne una API de encaminamiento basada en clave (*key-based routing, KBR*), con las siguientes funciones:

- *void route(key k, msg m).* Esta operación envía un mensaje, *m*, hacia el nodo responsable de la clave *k*. Esta función está implementada en la capa *overlay*.
- *void forward(key k, msg m, nodehandle nexthopnode).* Esta función está implementada en la capa de aplicación. Es invocada desde la capa *overlay* de cada nodo que envíe un mensaje *m* durante su encaminamiento.
- *void deliver(key k, msg m).* Esta función también está implementada en la capa de aplicación. Es invocada desde la capa *overlay* del nodo responsable de la clave *k* tras la llegada de un mensaje *m*.

Las redes *overlay* estructuradas se emplean con frecuencia para construir servicios de tabla hash (*hashtable*) distribuidos (DHT) en los que se almacena el mapeo entre una clave y un valor. Esta interfaz implementa una funcionalidad simple de almacenamiento y recuperación, dónde el valor es almacenado siempre en los nodos vivos de la red *overlay* a los que la clave está mapeada por la capa KBR. La interacción con el servicio DHT se realiza por medio de los mensajes de petición *Put* (publica un objeto en el servicio DHT) y *Get* (obtiene el objeto asociado a una clave específica), y los correspondientes mensajes de respuesta. Por tanto, los mensajes para interactuar con el servicio DHT proporcionado

por una red estructurada son *DHTRequestMessage* y *DHTResponseMessage*, con dos tipos diferentes, uno para el mensaje *Put* y otro para el mensaje *Get*.

OverSim ofrece un servicio DHT incorporado. Sin embargo, este servicio está basado en replicación y no es útil para muchas aplicaciones. En otras muchas situaciones, todo nodo almacena únicamente los valores asociados a las claves que se encuentran en su rango de responsabilidad. Sin embargo, en este caso, el software P2P estructurado de cada nodo debe notificar a la aplicación sobre los cambios en el conjunto de claves de las que el nodo es responsable. Esto permite al software de la aplicación, por ejemplo, mover los valores hasta sus nuevas ubicaciones cuando un nodo se une.

5.2. Red P2P basada en *Super-Peer*

Las operaciones comunes en una red P2P basada en *SP* son las siguientes:

- *SP joint*: El cliente se une a un *SP* específico enviándole un mensaje de petición. El *SP* responde con un mensaje de respuesta, indicando si se permite la conexión.

- *Publicación*: El cliente publica su lista de archivos compartidos enviando un mensaje de petición al *SP*. Éste responde con un mensaje de respuesta, indicando si las operaciones se han realizado correctamente.

- *Lookup*: El cliente busca los contenidos que desea descargar enviando un mensaje de petición al *SP*. Éste responde con un mensaje de respuesta, y el cliente obtiene una lista de clientes que tienen el archivo que éste desea.

- *Descarga*: Un cliente se conecta a otro cliente (una fuente) con el fin de descargar un archivo, enviando un mensaje de petición. El otro nodo responde entonces con un mensaje sobre las características de la conexión.

Por tanto, podemos establecer que las operaciones para interactuar con una red P2P basada en *SP* vienen dadas por:

- *Comunicación con un SP*: Esta comunicación se realiza con el mensaje *SuperPeerRequestMessage* y el correspondiente *SuperPeerResponseMessage*, con tipos diferentes: unión a *SP*, publicación de contenidos y búsqueda de contenidos.

- *Comunicación con un Peer*: Esta comunicación es realizada mediante el mensaje *PeerRequestMessage* y su correspondiente *PeerResponseMessage*. Estos mensajes son utilizados para iniciar la descarga del contenido requerido.

6. Descripción del modelo

6.1. Arquitectura de la Red Overlay

Se tomó el trabajo publicado en [2] como protocolo de referencia para ilustrar el comportamiento en simulación porque implementa el modo de interacción con una red P2P estructurada y una red basada en *Super-Peer* (*SP*). En esta propuesta, los *peers* se agrupan en subgrupos, formando una red P2P basada en *SP*. Sin embargo, esta topología es gestionada mediante un mecanismo de búsqueda estructurado.

Para buscar un contenido, el usuario enviará los parámetros de búsqueda a su *SP* local y éste resolverá la petición. Para permitir a los *SP*s la localización de contenidos ubicados en otros subgrupos remotos, todos los *SP*s de la red son miembros de un grupo *multicast*. La red P2P estructurada subyacente se utiliza para implementar un servicio de *Multicast* a Nivel de Aplicación (*Application Level Multicast, ALM*). De entre todas las posibilidades, se ha seleccionado Chord-multicast [10].

Por otro lado, todos los nodos están inmersos en una red estructurada Chord [11]. Por tanto, todo nodo tiene un identificador (*NodeID*), y tienen que contactar con un nodo existente para unirse a esta red. El nodo previamente existente también indica al nuevo nodo el identificador del subgrupo al que pertenece (*SubgroupID*), y éste último encontrará al *SP* de dicho subgrupo empleando el mecanismo de búsqueda estructurado. Esto es posible porque cada vez que un nodo se convierte en un *SP*, éste debe contactar con el nodo cuyo *NodeID* coincide con el *SubgroupID* y envía su propia dirección IP.

El modo de conexión es el siguiente: inicialmente, el nuevo nodo tratará de conectarse al mismo subgrupo que el nodo existente. Sin embargo, si no hay espacio para éste, se solicitará al nuevo nodo que cree un nuevo subgrupo (generado aleatoriamente) o se le pedirá que se una al subgrupo que el *SP* solicitado le instó a crear previamente. Cuando un nodo encuentra su *SP*, notifica sus recursos de ancho de banda y CPU. En función de estos parámetros, el *SP* formará una lista ordenada con los futuros mejores candidatos a *SP*. Esta lista es transmitida a todos los miembros del subgrupo.

6.2. Interacción con la red estructurada

Como ya se ha dicho anteriormente, nuestra aplicación utiliza el *SubgroupID* como clave y la IP de los *SuperPeers* como valor, y ambos se organizan en una DHT sobre una red estructurada Chord.

En entornos con direcciones IP dinámicas, cada vez que la dirección IP de un *SuperPeer* cambia, es necesario actualizar la información disponible en la DHT. El Apéndice A muestra el procedimiento empleado para llamar a la

función predefinida *route()* desde el nivel de aplicación, y la implementación de las funciones *forward()* y *deliver()* en OverSim.

6.3. Interacción con la red basada en Super-Peer

Cuando la capa *overlay* proporciona las direcciones IP de los *peers* en nuestra aplicación, la comunicación entre los nodos se establece por medio de un socket UDP/IP. Esta actividad requiere dos acciones. En primer lugar, los nodos tienen que asociar un puerto local para permitir la recepción de paquetes (función *bindToPort()*); entonces, con el fin de enviar mensajes a un nodo destino específico (*destAddr*, *destPort*) emplearemos la función *sendToUDP()* (Apéndice B).

El Apéndice C muestra la implementación de la recepción de un mensaje *SuperPeerRequestMessage* de tipo 1. Esta implementación requiere añadir las opciones apropiadas a la implementación de *handleMessage()*.

6.4. Resultados del modelo de simulación

En esta sección se presentan los resultados de simulación. Por motivos de espacio, en este trabajo sólo se presentan los resultados de un escenario denominado estático en el que los *peers* nunca mueren.

Los parámetros de la red son: 2,500 *peers*, 25,000 archivos y 50 *Super-Peers*. La simulación implica solamente a 2,500 *peers* con el fin de reducir el tiempo de simulación, aunque el entorno de simulación puede alcanzar fácilmente los 100,000 *peers*.

En primer lugar se presenta el comportamiento del sistema P2P con respecto a la popularidad de las consultas. Es bien sabido que el número de consultas emitidas para solicitar un archivo específico está conectado directamente con su popularidad. La popularidad asignada a toda consulta sigue una distribución Zipf, por tanto es razonable que el número de consultas emitidas durante una simulación tiene que seguir de algún modo la distribución de popularidad Zipf. Esta hipótesis se ve confirmada en los resultados presentados en la Figura 2 (en escala logarítmica).

La Figura 3 muestra la probabilidad de que un contenido se encuentre en el propio subgrupo del solicitante. Se observa que esta probabilidad crece conforme incrementa el tiempo, convergiendo a uno. Por tanto el sistema alcanza un estado estacionario. Esto también indica que con los típicos parámetros que modelan el comportamiento de las consultas, nuestro sistema asegura que los contenidos estarán distribuidos por igual entre todos los subgrupos en un mes.

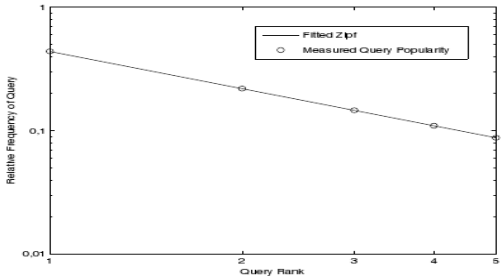

Figura 2: Popularidad de las consultas.

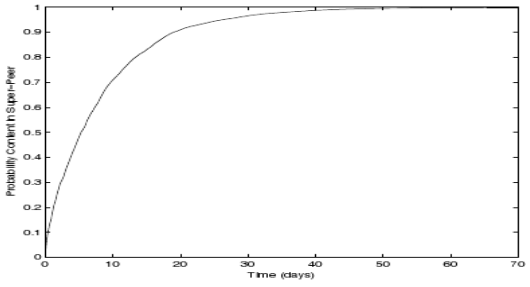

Figura 3: Probabilidad de que el contenido se encuentre en el *Super-Peer*.

7. Conclusiones

En este artículo se ha presentado una metodología para simular cualquier tipo de aplicación P2P empleando el entorno de simulación OverSim. Para ayudar al lector a comprender los principales conceptos principales, se ha utilizado una red overlay real, publicada previamente por los autores en otros trabajos. Las partes principales del código de simulación se muestran en lenguaje C++ real, aunque simplificado (ver apéndices).

De entre todas las herramientas disponibles se eligió OverSim debido a su flexibilidad. Su diseño modular y la utilización de una API común. Su diseño modular y el uso de la API común facilita el desarrollo de nuevos protocolos. Además, su flexible esquema de red subyacente puede ser muy útil durante el desarrollo de la aplicación. Asimismo, se han presentado y discutido figuras relacionadas con el rendimiento global de la simulación. Éstas revelan que la metodología expuesta puede ser utilizada y aplicada fácilmente por la comunidad investigadora para obtener resultados útiles e interesantes.

8. Agradecimientos

Esta investigación ha sido subvencionada por el proyecto TEC2007-67966-C03-01/TCM (CON-PARTE-1) y ha sido desarrollada en el marco de trabajo del "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia, de la Fundación Séneca, Agencia de Ciencia y Tecnología de la RM (Plan Regional de Ciencia y Tecnología 2007/2010)".

Antonio M. Martínez Rojo también agradece a la "Fundación Séneca" por una beca de investigación pre-doctoral asociada al proyecto "FORMA (Formación de personal cualifcado y herramientas y procesos telemáticos de valor añadido en el ámbito del Supercomputador Ben Arabí de la Región de Murcia)".

Referencias

[1] R. Steinmetz and K. Wehrle, Eds., Peer-to-Peer Systems and Applications, ser. Lecture Notes in Computer Science, vol. 3485. Springer, 2005.

[2] J. P. Muñoz-Gea, J. Malgosa-Sanahuja, P. Manzanares-Lopez, J. C. Sanchez-Aarnoutse, and A. M. Guirado-Puerta, A hybrid topology architecture for p2p file sharing systems, in Proceedings of the 1st International Conference on Software and Data technologies (ICSOFT 2006), Setúbal, Portugal, September 2006.

[3] P2psim: A simulator for peer-to-peer protocols," `http://pdos.csail.mit.edu/p2psim/`, 2005.

[4] Peersim: A peer-to-peer simulator, `http://peersim.sourceforge.net/`, 2006.

[5] Planetsim: Object oriented simulation framework for overlay networks, `http://planet.urv.es/trac/planetsim/`, 2005.

[6] "The oversim p2p simulator," `http://www.oversim.org/`, 2007.

[7] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers, The state of peer-to-peer simulators and simulations, SIGCOMM Comput. Commun. Rev., vol. 37, no. 2, pp. 95–98, 2007.

[8] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, Towards a common api for structured peer-to-peer overlays,in Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03), Berkeley, CA, February 2003.

[9] I. Baumgart, B. Heep, and S. Krause, 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.

[10] S. El-Ansary, L. O. Alima, P. Brand, and S. Haridi, Efficient broadcast in structured p2p networks, in IPTPS, 2003, pp. 304–314.

[11] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Trans. Netw., vol. 11, no. 1, pp. 17–32, 2003.

[12] J.P. Muñoz-Gea, J.M. Malgosa-Sanahuja, P. Manzanares-López, J. C. Sánchez-Aarnoutse, A.M. Martínez-Rojo: Simulation of a P2P Application Using Oversim, AFIN09: The First International Conference on Advances in Future Internet. Athens/Glyfada, Greece. ISBN: 978-0-7695-3664-4.