INASim: Una herramienta para la evaluación de redes de interconexión de altas prestaciones

Jesus Escudero-Sahuquillo, Pedro J. Garcia y Francisco J. Quiles[1]

Resumen—**El gran objetivo de la década en supercomputación es alcanzar los niveles de rendimiento en *exaflops* antes del año 2018. La red de interconexión es un componente esencial en los grandes sistemas de cómputo, debiendo ofrecer un gran ancho de banda y una latencia reducida: si la red es el cuello de botella del sistema, limitará las prestaciones de todo el supercomputador. En este sentido, la investigación en redes de interconexión se orienta principalmente a aumentar sus prestaciones básicas, pero también, y especialmente en el caso de redes para supercomputadores *exascale*, a reducir su consumo, proporcionar tolerancia a fallos y/o calidad de servicio o controlar la congestión. Para el diseño de las nuevas arquitecturas de red de interconexión, los simuladores son una herramienta crucial, ya que permiten modelar la arquitectura de red en la que estamos investigando, sin necesidad de invertir en hardware. En este artículo presentamos INASim, un simulador de redes de interconexión de propósito general, apto para modelar las arquitecturas de red del futuro, gracias a su flexibilidad en las topologías de red permitidas, a su extensibilidad para futuros modelos de red y su capacidad para simular redes de interconexión de gran tamaño. Además, INASim permite la evaluación de prestaciones de las redes simuladas usando trazas de tráfico real de los *benchmarks* más representativos.**

Palabras clave— **Redes de interconexión de altas prestaciones, Evaluación de prestaciones, Herramientas de simulación**

I. Introducción

El gran objetivo de la década en los sistemas de supercomputación es conseguir un rendimiento a nivel de *exaflops* antes del año 2018. En este sentido, se espera un crecimiento el el número de nodos de los supercomputadores mientras el consumo de potencia se mantiene. La red de interconexión [1] es un componente fundamental en los supercomputadores y el objetivo principal de los diseñadores ha sido, es y será que la red de interconexión ofrezca unas altas prestaciones y una baja latencia, para no ser el cuello de botella del supercomputador. En cuanto a tecnologías de red de interconexión, se puede observar en un vistazo a la clasificación de los supercomputadores más potentes del mundo (TOP500 [2]) que Infiniband [3] o Myrinet [4] están presentes en los actuales sistemas, y que otro gran porcentaje de tecnologías de red son o bien propietarias o bien diseñadas a propósito para los supercomputadores de los que forman parte. Además, un elevado porcentaje de sistemas son de tipo *cluster*. Por otro lado, de un tiempo a esta parte la reducción del coste y el consumo de las redes de interconexión es de una necesidad, ya que el consumo de potencia de la red supone más de un 30 % del total del sistema de supercomputación [5].

Específicamente, el rendimiento de la red de interconexión depende de la topología, que debe maximizar el ancho de banda ofrecido a los nodos interconectados; del algoritmo de encaminamiento elegido para discernir qué porcentaje de tráfico encamina por una u otra ruta de la red; de la política de tolerancia a fallos que se implemente en el sistema para recuperarse ante un eventual fallo, de los mecanismos de control de congestión que deben eliminar sus efectos negativos, etc. Y todas estas tareas deben tener en cuenta el requisito esencial de requerir un bajo consumo de potencia.

Durante la "*era exascale*"[1], se están replanteando los diseños arquitectónicos de todos los componentes de un supercomputador, entre ellos la red de interconexión, por lo que los diseñadores de redes de interconexión necesitarán herramientas que los asistan en el diseño abstracto de estas redes, tales como simuladores o herramientas de elaboración de prototipos, entre otras. Es evidente que los diseñadores de redes de interconexión necesitan abstraer las nuevas arquitecturas de red de interconexión para modelar las nuevas topologías, algoritmos de encaminamiento, estrategias de control de congestión y consumo, etc., ya que los nuevos diseños no tienen porqué estar basados en los diseños actuales. De este modo, las herramientas de simulación, que han sido esenciales en el pasado para el diseño de las redes de interconexión actuales, también lo son para el diseño de las nuevas arquitecturas de red que interconectarán a miles y miles de nodos de procesamiento.

Existe una gran variedad de herramientas de simulación de redes de interconexión disponibles bajo licencias de código abierto o comerciales, que se pueden usar para modelar redes de interconexión. El problema de muchas ellas es que ofrecen la posibilidad de modelar con mucho detalle las redes de interconexión, y esto limita que se puedan modelar grandes redes de interconexión. Otra tendencia en el diseño de herramientas de simulación de redes de interconexión es diseñar soluciones "al uso" que modelan aspectos muy específicos de ciertas arquitecturas o tecnologías de red. Además, en muchos casos, estas soluciones no son extensibles, puesto que cuesta mucho trabajo añadir el modelado de nuevas características de la red; su desarrollo y soporte se abandonan a corto plazo. En cierto modo, esta última práctica es muy común en investigación lo que supone muchas horas de trabajo invertidas en desarrollar una herramienta que luego será abandonada.

En este artículo proponemos INASim (del inglés *Interconnection-Networks Agile-Simulator*), una herramienta de simulación de propósito general orientada al modelado de forma eficiente y ágil de redes de interconexión de altas prestaciones con un número elevado de nodos. INASim es extensible, puesto que permite modelar nuevos aspectos de la red de interconexión con un esfuerzo mínimo. Su arquitectura es modular para minimizar el impacto de los nuevos aspectos de la red que pudieran modelarse en el futuro. INASim permite la automatización de simulaciones en un sistema *cluster* y la recogida de resultados de forma sencilla y rápida. En este sentido, el proceso de desarrollo de INASim (iterativo, incremental y ágil), ha permitido recientemente que varios grupos de investigación a nivel europeo se hayan implicado en su

[1]Dpto. de Sistemas Informáticos, Universidad de Castilla-La Mancha, e-mail: `jesus.escudero@uclm.es`.

[1]Término empleado para referirse al reto de conseguir niveles de rendimiento en *exaflops* durante este década

desarrollo. En la actualidad, INASim está siendo traducido a un modelo en OMNeT++ [6], lo que supone un paso en la apertura de este sistema a la comunidad científica y un avance en la validación y verificación del modelado de redes de interconexión.

El resto de este artículo se organiza como sigue. La sección II describe y analiza las propuestas de herramientas de simulación más representativas. La sección III describe la herramienta (arquitectura, modelado, automatización de simulaciones y métricas). La sección IV muestra los resultados obtenidos para ciertas topologías modeladas por INASim. Finalmente, la sección V discute los resultados más importantes de este trabajo.

II. Trabajo Relacionado

La simulación es la imitación del comportamiento de un ente a lo largo del tiempo [7], y se usa en muchos campos de la investigación como la automoción, aeronáutica, modelado del tráfico de las carreteras, mecanismos de seguridad en los vehículos a motor, finanzas, sociología, etc.

En el campo de la supercomputación, es muy común el uso de simuladores para el modelado de redes de interconexión de altas prestaciones. En este sentido, existen numerosas soluciones de propósito específico tanto a nivel académico como comercial. Además, hay plataformas de simulación de propósito general que también son muy populares.

En cuanto a plataformas de propósito general se podrían destacar varias de ellas. Por ejemplo, NS-2 y NS-3 [8] están principalmente dirigidos al mundo de la investigación en redes de comunicaciones, siendo muy usados en redes de móviles. OMNeT++ [6] es una plataforma de simulación en auge, con una potente interfaz gráfica que facilita la depuración de modelos y un diseño modular. OMNeT++ es código abierto, al igual que NS-3, y tiene un gran prestigio en la investigación académica. Otra opción, dentro del ámbito comercial es OPNET [9] que se usa ampliamente tanto en los entornos académicos como en las empresas privadas. El problema de OPNET es que las simulaciones requieren mucha potencia de cómputo, que se incrementa de forma significativa con el tamaño del sistema. Esto es debido al alto nivel de detalle alcanzado por OPNET para modelar los componentes del sistema. Una solución podría ser usar nuevos componentes de “grano gordo”, pero a costa de invertir más tiempo en su modelado.

Por el contrario, existen herramientas de propósito más específico más compactas y sencillas para el modelado de arquitecturas específicas de redes de interconexión. Un ejemplo de herramienta de código abierto como Booksim [10], inicialmente planteado para redes *off-chip*, pero recientemente reformado para redes *on-chip*. Este simulador, aunque es muy simple, tiene limitaciones en cuanto a las opciones de modelado. Por ejemplo sólo modela patrones de tráfico sintético. Por otro lado, INSEE [11] es un simulador escrito en ANSI C, orientado a simular de forma eficiente redes de un gran número de nodos. Además, permite numerosas topologías de red, algoritmos de encaminamiento, tolerancia a fallos, etc. INSEE también modela tráfico real mediante el uso de trazas y se puede comunicar con instancias de SIMICs para que éstas dirijan la ejecución de la simulación. Según sus propios desarrolladores, no se modelan con detalle tecnologías de red como Infiniband [3] o Myrinet [4], porque ese nivel de detalle incrementaría el consumo de recursos en tiempo de ejecución del simulador. Otra propuesta representativa es la herramienta TOPAZ [12], que incluso incrementa el número tamaño de la red a simular, alcanzando el millón de encaminadores. TOPAZ modela principalmente redes directas y abarca tanto redes *on-chip*, como *off-chip*. TOPAZ también ha sido paralelizado y esto le permite acelerar el tiempo de ejecución. Al igual que INSEE o BookSim no se centra en el modelado de tecnologías de red actuales.

De este breve recorrido por algunas herramientas de simulación se desprende que lo ideal sería desarrollar herramientas de simulación que cumplieran las siguientes características:

- **Escalabilidad**. Posibilidad de modelar redes de gran tamaño de forma eficiente, sin que el tiempo de ejecución y los recursos de la herramienta sean excesivos.
- **Rapidez**. El tiempo de ejecución de las simulaciones debe ser tan reducido como sea posible.
- **Extensibilidad**. Posibilidad de añadir nuevos módulos que modelan nuevas arquitecturas o ciertas partes de las mismas, sin la necesidad de invertir mucho tiempo en la programación.
- **Modularidad**. Sería deseable que un desarrollador sólo modificara un subconjunto reducido de los módulos de la herramienta para aplicar un cambio o añadir un nuevo diseño. Esta modularidad permite que el impacto de un cambio en la herramienta sea mínimo y así reducir al mínimo las tareas de depuración, validación y verificación.
- **Nivel de detalle**. Aunque es deseable la escalabilidad, el simulador debería modelar con cierto nivel de detalle todas partes de la pila de un supercomputador de altas prestaciones; desde la capa de aplicación y los nodos de procesamiento, hasta las tecnologías de red y la arquitectura del conmutador, manteniendo un equilibrio entre sencillez y nivel de detalle.

En este sentido, en la siguiente sección se describe la herramienta INASim cuya filosofía intenta satisfacer las características que acabamos de describir, deseables a una herramienta de simulación. La herramienta INASim ha servido para la evaluación de prestaciones de las técnicas desarrolladas en las tesis doctorales [13], [14].

III. Descripción de INASim

INASim (*Interconnection-Networks Agile-Simulator*) es una herramienta de simulación de redes de interconexión de altas prestaciones utilizadas en los grandes sistemas de supercomputación, dirigido por eventos que se suceden en el tiempo de ejecución. Está escrito en C++ y se ha modelado siguiendo un paradigma de programación orientado a objetos. INASim modela redes de un tamaño elevado (ha sido testeado para 32.000 nodos de procesamiento) y es capaz de gestionar cargas de tráfico muy elevadas sin que se eleve demasiado su consumo de recursos durante el tiempo de ejecución. Su diseño extensible y modular está pensado para que la inversión de tiempo en su desarrollo sea mínima, a la hora de introducir nuevas características a los modelos de red.

Por otro lado, INASim modela las arquitecturas de red

con cierto nivel de detalle (el necesario según la configuración de la red). Por ejemplo, para evaluar estrategias de control de congestión en las redes de interconexión, INASim modela las partes del control de congestión de la especificación de Infiniband, pero sólo las características relativas al control de congestión y no la especificación Infiniband entera.

A continuación, se describe la arquitectura de INASim, cómo se ha modelado la funcionalidad principal de la herramienta, se describe brevemente el código fuente y cómo se han modelado los diferentes patrones de tráfico.

A. Arquitectura General

La figura 1 muestra un diagrama de la arquitectura del simulador INASim. Se puede observar que INASim recibe como entrada un fichero de parámetros en formato XML, realiza un análisis de esos parámetros usando el módulo *XML Parser* y obtiene una configuración del sistema y una configuración del motor de comportamiento. Con esa información INASim construye el sistema de HPC (*High-Performance Computing -HPC- System*).

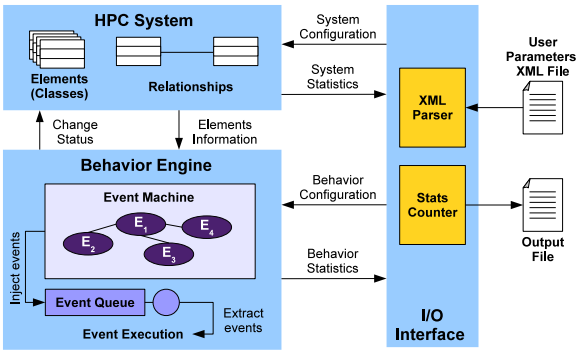


Fig. 1. Arquitectura del simulador INASim.

Por otro lado, la parte fundamental del simulador es el motor de comportamiento (*Behavior Engine*) que modela la máquina de eventos y la cola de eventos. La máquina de eventos se encarga de introducir eventos en la cola de eventos, en función de las dependencias previamente establecidas entre los eventos y la información de los elementos del sistema HPC. Es importante destacar que dependiendo de la información recibida en el fichero de parámetros INASim modifica la máquina de eventos.

Téngase en cuenta que INASim permite el modelado de la máquina de eventos del sistema para definir los comportamientos deseados de cualquier sistema HPC. Obviamente, debe haber un conocimiento del sistema HPC que se ha modelado porque los eventos modifican los elementos del sistema HPC y éstos últimos dan información a la máquina de eventos para tomar decisiones de comportamiento. No obstante, esta separación entre sistema HPC y motor de comportamiento no es vana, sino que permite un alto grado de reutilización del código destinado al comportamiento. Por tanto, no será necesario modificar todo el código del comportamiento del sistema si se incluyen nuevas partes al sistema HPC y, inversamente, no será obligatorio modificar el el código del sistema HPC cuando se modifique el comportamiento.

Finalmente, tanto el sistema HPC como al motor de comportamiento generan estadísticas que son recogidas por el módulo *Stats Counter* que se encargará de generar los ficheros de resultados correspondientes a una simulación concreta. En la siguiente sección se profundiza un poco más en la arquitectura de la herramienta INASim.

B. Modelado del Sistema

La figura 2 muestra, mediante un diagrama de clases UML, las clases más representativas de la arquitectura de INASim. Se observan las tres partes de la arquitectura de INASim de la figura 1: Entrada/Salida, Motor de Comportamiento y Sistema de HPC.

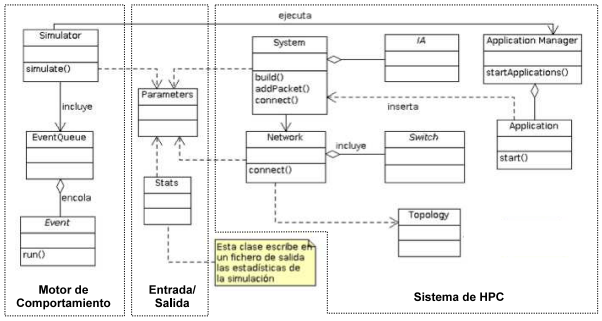


Fig. 2. Diagrama de clases general.

En concreto, dentro del la Entrada/Salida, la clase *Parameters* se encarga de leer el fichero XML recibido y guardar la configuración del sistema de forma global durante todo el tiempo de la simulación. Todos los objetos del sistema de HPC se construyen con la información de esta clase. Por otro lado, la clase *Stats* se encarga de recopilar la información de la simulación y guardarla en un fichero de salida. Todas las métricas que ofrece INASim (por ejemplo la productividad de la red, la latencia de red, el consumo de recursos, información de eventos, carga generada por el sistema de cómputo, etc.) se calculan por medio de esta clase; y toda esa información se envía a los diferentes ficheros de salida.

Por su parte, el Motor de comportamiento está formado por la clase *Simulator* que básicamente construye el tanto la cola de eventos (clase *EventQueue*), como el motor de comportamiento (clase *Event*). Es importante destacar que la clase *Event* es abstracta y de ella extenderán todos los eventos que forman la máquina de eventos. Tal y como establece la filosofía de los simuladores dirigidos por eventos, unos eventos dispararán otros eventos y estos a su vez a otros, y así sucesivamente. Como se ha descrito, INASim permite definir la máquina de eventos propia de un sistema HPC, así como el nivel de detalle con el que unos eventos disparan a otros. Por ejemplo, el comportamiento de un *buffer* de un conmutador se puede modelar mediante dos eventos: uno de inserción y otro de extracción de paquetes (este sería el modo más simple); o bien se puede modelar también desgranando los eventos de inseción/extracción en eventos más específicos que aplicaran políticas de fragmentación de la memoria SRAM que implementa ese *buffer*. Obviamente, la segunda opción requerirá más tiempo de cómputo para la ejecución de la simulación.

Por su parte, INASim modela el sistema de HPC mediante la capa de aplicación, los nodos de procesamiento y la red de interconexión. La capa de aplicación (*ApplicationManager*) modela el tráfico que será generado en el sistema de HPC. INASim es capaz de ejecutar varias aplicaciones (*Application*) dentro de un mismo nodo de procesamiento, tal y como ocurre en los sistemas

de cómputo reales. En concreto, INASim modela tráfico sintético (uniforme, perfect-shuffle, bit-reversal, etc.) y tráfico real, mediante el uso de trazas de *benchmarks* ampliamente usados en la evaluación de sistemas HPC (LinPack, NAS, P-TRANS, Worf, Gromacs, etc.).

En consecuencia, cada *Application* en ejecución genera mensajes dentro de un sistema HPC, que se modela en INASim mediante la clase *System*. Básicamente, dicha clase está formada por un conjunto de adaptadores de red (*IAs*), encargados de inyectar en la red los mensajes generados por la clase *Application*, y por la propia red de interconexión, que se modela mediante la clase *Network*. La red de interconexión se construye con la información aportada por la clase *Topology*, encargada de definir el patrón de conectividad entre los conmutadores de la red (clase *Switch*) y las conexiones de dichos conmutadores con los nodos de procesamiento, por medio de los *IAs*. Además, la clase *Topology* se encarga aportar la información pertinente a la lógica de encaminamiento de cada conmutador. En la siguiente sección se describe de manera más específica cómo se han modelado los conmutadores.

C. Modelado del conmutador

El modelo de conmutador de INASim se basa en una micro-arquitectura segmentada en varias etapas funcionales [10], [15]. La figura 3 muestra un diagrama de esta micro-arquitectura, perteneciente a un conmutador clásico. Por un lado, se observa en la parte superior el camino de los datos (*Internal Data Path*) seguido por la porción de datos de cada paquete que cruza el conmutador. Por otro lado, se observa el plano de control, donde se toman las decisiones de encaminamiento, arbitraje y control de flujo en función de la información contenida en la cabecera de cada paquete.

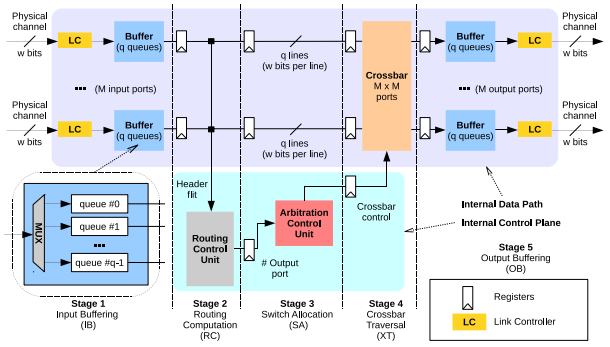


Fig. 3. Micro-arquitectura del conmutador.

Por tanto, para cada paquete que llega a un conmutador (etapa IB=*Input Buffering*), esta organización separa la cabecera y los datos. La cabecera viaja por las unidades funcionales de las etapas del plano de control, donde se toman las decisiones de encaminamiento (etapa RC=*Routing Computation*) y arbitraje (etapa SA=*Switch Allocation*). En cambio los datos viajan por las unidades funcionales del camino de los datos (es decir en las etapas IB=Input Buffering, XT=Crossbar Traversal, OB=Output Buffering); esperando en los registros intermedios a que se tomen las decisiones de control. Es importante destacar que en la etapa OB la cabecera y los datos se unen, y el paquete se envía por el enlace hacia el conmutador siguiente o el nodo de procesamiento destino.

Esta arquitectura de conmutador permite modelar y modular de forma de independiente las diferentes unidades funcionales del conmutador. De esta forma, se también se simplifica el diseño de cambios futuros en estas unidades funcionales. Además, este diseño permite modelar con gran precisión la latencia de conmutador que vendrá definida por el tiempo que la cabecera estará retenida en las unidades funcionales de control. La figura 4 muestra la latencia del conmutador para un paquete en ausencia de contención dentro del conmutador.

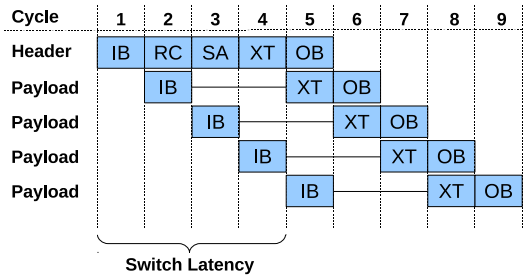


Fig. 4. Etapas de la segmentación del conmutador.

A continuación se describe cómo se ha modelado en INASim la micro-arquitectura descrita. Téngase en cuenta que estamos describiendo únicamente el sistema y no el comportamiento del mismo. Como se ha descrito el comportamiento viene definido por la máquina de eventos. La figura 5 muestra las clases que modelan la micro-arquitectura del conmutador.

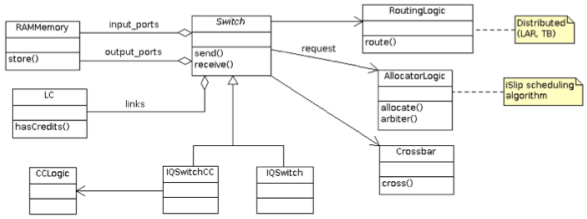


Fig. 5. Diagrama de clases y relaciones de la clase Switch.

En concreto, la clase *Switch* está formada por un conjunto de unidades funcionales. En primer lugar, dicha clase contiene un conjunto de *buffers* tanto en los puertos de entrada, como en los puertos de salida, que se modelan con la clase *RAMMemory*. Cada puerto del conmutador se modela con la clase *LC* (controlador de enlace), que modela un extremo del enlace segmentado bidireccional. A su vez la lógica de encaminamiento se modela mediante la clase *RoutingLogic*, que se encarga de calcular la información de encaminamiento para cada paquete. Por su parte la clase *AllocatorLogic* modela la generación de peticiones de cruce de cada paquete dentro del conmutador, e implementa un algoritmo de arbitraje para elegir qué peticiones de entre todas las generadas se eligen para que sus paquetes correspondientes se envíen al puerto de salida correspondiente. Las conexiones entre puertos de entrada y salida se modelan mediante un *crossbar* y su clase asociada. Básicamente, la clase *crossbar* indica qué conexiones están disponibles para las peticiones seleccionadas.

Finalmente, la clase *Switch* heredan la clase *IQSwitch*, que modela un conmutador con *buffers* sólo en los puertos de entrada (conmutador IQ), y la clase *IQSwitchCC*, que modela un conmutador IQ que incluye una lógica de

TABLA I

PRINCIPALES PARÁMETROS CONFIGURABLES DE INASIM

Funcionalidad	Opciones
Topologías	Indirectas (fat-trees, shuffle-exchange, baseline, etc.), Directas (mallas, toros, hipercubos, etc.) y configurables
Encaminamiento	Determinista (DESTRO, XY) Adaptativo (en desarrollo)
Conmutación	VCT, Packet-switching
Control de Flujo	Créditos, Burbuja [16]
Arbitraje	Round-Robin, FIFO, iSlip [17]
Arquitectura de Conmutador	Input-Queued Segmentado
Organización Buffers	VCs [18], VOQsw [19], VOQnet [20], DAMQs [21]
Tráfico	Sintético (Uniforme, Perfect-Shuffle, Bit-Reversal, etc.) y basado en trazas (LinPack, NAS, P-TRANS, Worf, Gromacs, etc.)

control de congestión. Esta lógica se ha usado para modelar las estrategias de control de congestión definidas en [13]. La tabla I muestra los parámetros básicos de configuración de INASim.

IV. EJEMPLOS DE SIMULACIONES

En esta sección se muestran varios ejemplos de configuraciones de red usando INASim y su posterior evaluación de prestaciones. La tabla II muestra dos topologías de red multietapa bidireccionales para un sistema de 1024 nodos. En concreto un k-ary n-tree y un k-shuffle-exchange. Los patrones de tráfico sintético que se han modelado para estas dos configuracones de red se muestran en la tabla III. En concreto, el patrón 1 modela tráfico uniforme en función de la carga. Es decir, el porcentaje de tráfico generado se incrementa de manera incremental. En cambio, el patrón de tráfico sintético 2 modela la aparición de una situación de congestión eventual generada por el 25 % de los nodos en el intervalo de tiempo 1000-2000 μs. Además se asume que la arquitectura del conmutador usa *buffers* sólo en los puertos de entrada. En este sentido se han modelado varios esquemas de organización de dichos *buffers* (ver tabla I).

La figura 6 muestra los resultados obtenidos por INASim para las configuraciones de red de la tabla II y los patrones de tráfico de la tabla III.

Como se ha descrito previamente, INASim permite la generación de tráfico basado en trazas de tráfico real. La figura 7 muestra los resultados para dos configuraciones de red diferentes a las anteriores en cuanto al número de nodos.

V. CONCLUSIONES

Las herramientas de simulación son un elemento muy importante para la investigación y el diseño de las arquitecturas de red de interconexión de altas prestaciones de esta década. Antes del año 2018, se prevé que dichas arquitecturas interconecten miles de nodos, formando sistemas de computación que alcancen los niveles de rendimiento en *exaflops*.

Las numerosas herramientas de simulación propuestas pueden clasificarse en dos grupos que se ajustan a modelos opuestos. Por un lado, existen herramientas basadas en la simplicidad en el diseño de los componentes de la red con el objetivo de modelar redes de interconexión de gran tamaño, que interconectan miles de nodos. Por el contrario, otras herramientas intentan conseguir un gran nivel de detalle a la hora de modelar dichos componentes de red, lo que requiere mucha potencia de cómputo y un gran consumo de recursos, si incrementamos el tamaño de la red de interconexión; por tanto no son escalables con el tamaño de la red.

En este artículo se ha presentado la herramienta de simulación INASim, que establece un compromiso entre la escalabilidad y nivel de detalle en el modelado de las redes de interconexión del futuro. INASim es capaz de modelar diversas topologías de red que interconectan miles de nodos, a la vez que ofrece un gran nivel de detalle orientado a modelar tecnologías de red comerciales. Además, INASim es capaz de inyectar en la red tráfico real de los *benchmarks* más representativos para la evaluación de sistemas HPC, además de los habituales patrones de tráfico sintético.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por los antiguos MEC y MICIIN (ahora MINECO), así como por la Comisión Europea a través de los fondos FEDER, mediante los proyectos CSD2006-00046 y TIN2009-14475-C04. Además, también ha sido parcialmente financiado por la JCCM, mediante el proyecto POII10-0289-3724 y la beca de doctorado A08/048.

REFERENCIAS

[1] J. Duato, Sudhakar Yalamanchili, and Lionel Ni, *Interconnection Networks An Engineering Approach*, Morgan Kaufmann, revised edition edition, 2003.

[2] Top 500 List, ," Web Page at: `http://www.top500.org`.

[3] InfiniBand Trade Association, *InfiniBand architecture specification volume 1. Release 1.2.1*, Nov. 2007.

[4] Myrinet, 2000 Series Networking, ," Web Page at: `http://www.myricom.com`.

[5] The Advanced Scientific Computing Advisory Committee (ASCAC). Subcommittee on Exascale Computing, "The Opportunities and Challenges of Exascale Computing," Tech. Rep., 2010.

[6] Andrés Varga, "The OMNeT++ Discrete Event Simulation System," *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.

[7] *Discrete-Event System Simulation (4th Edition)*.

[8] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala, "Improving Simulation for Network Research," Tech. Rep. 99-702b, University of Southern California, March 1999, revised September 1999, to appear in IEEE Computer.

[9] Inc. OPNET Technologies, "OPNET Modeler," `http://www.opnet.com`.

[10] William Dally and Brian Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[11] Javier Navaridas, Jose Miguel-Alonso, Jose A. Pascual, and Francisco J. Ridruejo, "Simulating and evaluating interconnection networks with INSEE," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 494–515, 2011.

[12] Pablo Abad, Pablo Prieto, Lucia G. Menezo, Adrián Colaso, Valentin Puente, and Jose-Angel Gregorio, "TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers," *Networks-on-Chip, International Symposium on*, vol. 0, pp. 99–106, 2012.

[13] Jesus Escudero-Sahuquillo, *Towards Efficient and Feasible Congestion Management Techniques for High-Performance Interconnection Networks Using Distributed Routing*, Ph.D. thesis, University of Castilla-La Mancha, 2011.

[14] Juan Antonio Villar Ortiz, *High-Radix Combined Switches: Formalization and Configuration Methodology*, Ph.D. thesis, University of Castilla-La Mancha, April 2012.

[15] J. Duato and Timothy Pinkston, *Interconnection Networks*, Computer Architecture - A Quantitative Approach. Appendix E. Morgan Kaufmann, 4th edition, 2007.

[16] C. Carrión, R. Beivide, J. A. Gregorio, and F. Vallejo, "A Flow Control Mechanism to Avoid Message Deadlock in k-ary n-cube Networks," in *Proceedings of the Fourth International Conference on High-Performance Computing*, Washington, DC, USA, 1997, HIPC '97, pp. 322–, IEEE Computer Society.

TABLA II

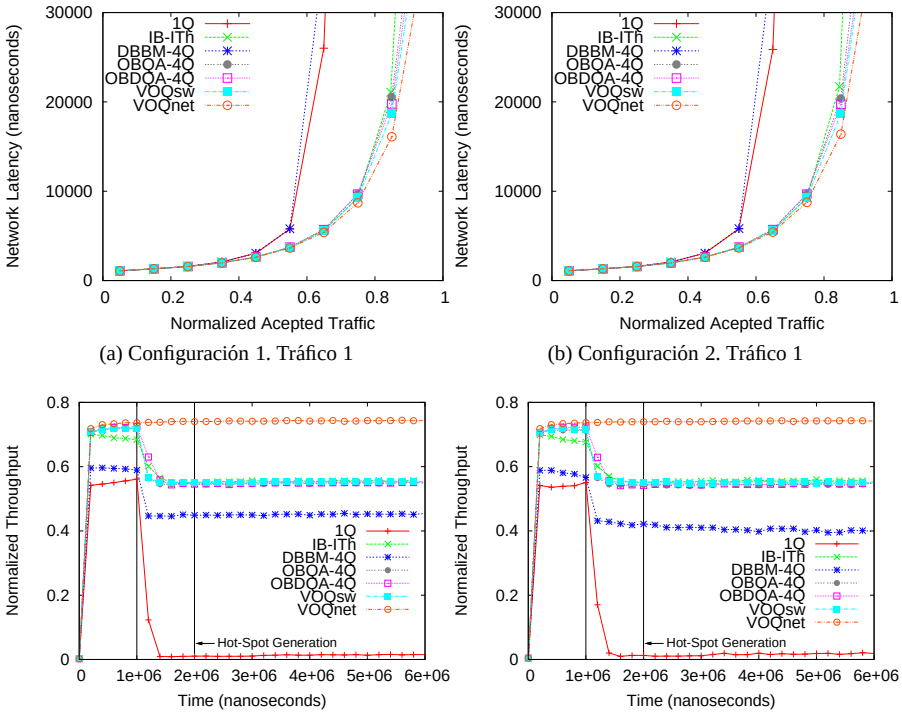
CONFIGURACIONES DE RED

#	Network Size	Interconnection Pattern	Switch radix	#Switches (total)	#Stages
1	1024 × 1024	4-ary 5-tree	8	1280	5
2	1024 × 1024	1024-node 4-shuffle-exchange	8	1280	5

TABLA III

PATRONES DE TRÁFICO SINTÉTICO

	Random Traffic			Hot-Spot Traffic				
#	% Sources	Destination	Generation rate	% Sources	Destination	Generation rate	Start time	End time
1	100 %	random	incremental	0 %	-	-	-	-
2	75 %	random	100 %	25 %	1 Fixed Dst. (1 Congestion Tree)	100 %	1000 μs	2000 μs



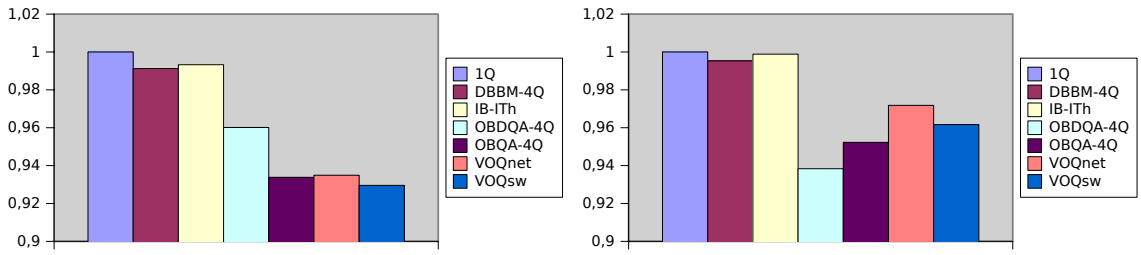
(a) Configuración 1. Tráfico 1

(b) Configuración 2. Tráfico 1

(c) Configuración 1. Tráfico 2

(d) Configuración 2. Tráfico 2

Fig. 6. Resultados obtenidos con INASim usando tráfico sintético.



(a) 4-ary 3-tree.

(b) k-shuffle-exchange.

Fig. 7. Tiempo de ejecución con tráfico LINPACK.

[17] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188–201, 1999.

[18] William Dally, "Virtual-Channel Flow Control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, pp. 194–205, 1992.

[19] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-Speed Switch Scheduling for Local-Area Networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319–352, November 1993.

[20] William Dally, P. Carvey, and L. Dennison, "Architecture of the Avici terabit switch/router," in *Proc. of 6th Hot Interconnects*, 1998, pp. 41–50.

[21] Y. Tamir and G.L. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches," *IEEE Transactions on Computers*, vol. 41, no. 6, June 1992.