Simulación de redes de interconexión utilizando tráfico real

F. J. Ridruejo Pérez, J. Miguel-Alonso[1]
Universidad del País Vasco UPV/EHU
Departamento de Arquitectura y Tecnología de Computadores
Apdo. 649, 20080 San Sebastián
{j.miguel, franciscojavier.ridruejo}@ehu.es

Resumen

La simulación de redes de interconexión para sistemas paralelos y distribuidos requiere de mecanismos para la generación de tráfico. Este tráfico puede ser sintético, extraído de trazas, u obtenido a partir de aplicaciones paralelas en ejecución. En este trabajo describimos cómo, en el contexto de TrGen (el módulo de generación de tráfico de INSEE) estamos usando un sistema de simulación conducida por la ejecución basado en SIMICS para obtener tráfico real y alimentar con él simuladores de redes de interconexión.

1. Introducción

En el diseño de un sistema paralelo, el sub-sistema de comunicación entre procesadores es un elemento crítico. Existe toda una línea de investigación en el área de redes de interconexión.

Muchos trabajos en esta área utilizan como elemento clave la simulación. Podemos encontrarnos, como software comercial o gratuito, multitud de simuladores de redes. Todos ellos tienen algún mecanismo para la generación de tráfico durante la simulación. Los patrones de tráfico empleados tienen una influencia trascendental en los resultados obtenidos. Lo ideal es que se usen tráficos reales, generados por las aplicaciones que van a ejecutarse en el sistema paralelo.

Hay diferentes maneras de generar el tráfico que alimenta a los simuladores. Ordenándolos de peor a mejor en cuanto a la fidelidad del tráfico generado (con respecto a las aplicaciones reales), tenemos:

- Patrones de tráfico sintéticos, obtenidos aplicando distintas distribuciones estadísticas. Ejemplos de ello son el uniforme, la matriz transpuesta, el *hot-spot* (punto caliente) o el *hot-region* (región caliente).
- Tráfico basado en trazas, que es obtenido mediante la generación de trazas durante la ejecución de las aplicaciones paralelas en un sistema real—normalmente, uno distinto al que se está estudiando.
- Simulación conducida por la ejecución, donde no sólo se simula la red de interconexión y el tráfico que la recorre, sino también los nodos conectados a ella. De esta manera se ejecuta la aplicación real en los nodos simulados que se comunican utilizando la red de interconexión simulada.

Como en otros muchos contextos, la fidelidad está reñida con la velocidad. Así, muchas veces se usan, en las primeras fases de evaluación de una propuesta de diseño, los patrones sintéticos, para refinar la propuesta y no continuar si no es prometedora. La generación de trazas es más compleja, porque pueden ser de gran tamaño y, además, se necesitan muchas: tantas como aplicaciones objetivo. La simulación conducida por ejecución sólo nos la podemos plantear cuando tenemos suficientes recursos, en forma de máquinas y tiempo, porque el grado de detalle de simulación exigido hace que sea muy costosa computacionalmente.

Nuestro grupo lleva unos años desarrollando INSEE [6], un entorno de evaluación de redes de interconexión, cuyos elementos principales son FSIN (un simulador funcional de redes) y TrGen [5] (una colección de módulos de generación de

[1] Este trabajo ha sido posible gracias a la financiación del Ministerio de Educación y Ciencia (TIN2004-07440-C02-02) y de la Diputación Foral de Gipuzkoa (OF-846/2004).

tráfico). Este artículo se centra en la parte de TrGen encargada del tráfico real, conducido por la ejecución. Se ha realizado utilizando SIMICS [2] como elemento de base para la simulación de los nodos de cómputo. La estructura modular de SIMICS nos ha permitido añadir elementos para extraer/insertar mensajes, que son procesados por un simulador de redes.

En la próxima sección describimos someramente INSEE, para poner este trabajo en contexto. La sección 3 contiene el grueso de este trabajo: los diseños realizados sobre SIMICS para integrarlos en nuestro entorno de simulación. En la sección 4 recogemos algunas conclusiones sobre el trabajo realizado.

2. INSEE

INSEE (*Interconnection Network Simulation and Evaluation Environment*) está organizado como un conjunto de módulos, muchos de los cuales se pueden utilizar como aplicaciones independientes. FSIN (*Functional Simulator of Interconnection Networks*) y TrGen (*Traffic Generator*) forman su núcleo.

En la Figura 1 se puede ver representado el diseño de INSEE. Los elementos sombreados en gris forman parte de este entorno y han sido desarrollados por nuestro grupo. Los demás son módulos externos pero pueden interactuar con él usando su API. SICOSYS [4] ha sido desarrollado independientemente, pero puede considerarse que forma parte de INSEE. En la parte de la derecha se encuentran los diferentes simuladores de redes de interconexión y a la izquierda están representadas las distintas fuentes de tráfico.

2.1. FSIN

FSIN es un simulador funcional que sirve para asistir en el diseño de redes de interconexión y para la ayuda en la toma de decisiones en las primeras fases de este diseño.

Se trata de un simulador muy flexible ya que se pueden controlar multitud de parámetros. Además, es muy rápido, porque no simula los detalles hardware, consumiendo muy poca memoria. Esto permite simular redes de gran tamaño (miles de nodos) con un equipo de sobremesa. El grupo de investigación ya contaba con un simulador de redes de interconexión, SICOSYS, pero éste detalla los elementos de la red a un nivel próximo a su implementación hardware, lo que lo hace dos órdenes de magnitud más lento y pesado en memoria. En cambio ofrece una información sobre tiempos mucho más detallada que FSIN.

FSIN simula encaminadores como el mostrado en la Figura 2. Algunos de los parámetros que se pueden cambiar son: tamaños de colas y búferes, número de dimensiones, número de canales virtuales por enlace físico, enlaces uni- o bi-direccionales, tamaño del paquete en phits, topología de la red, tamaño de la misma, estrategia de gestión, de petición y política de asignación de los canales virtuales, etc.

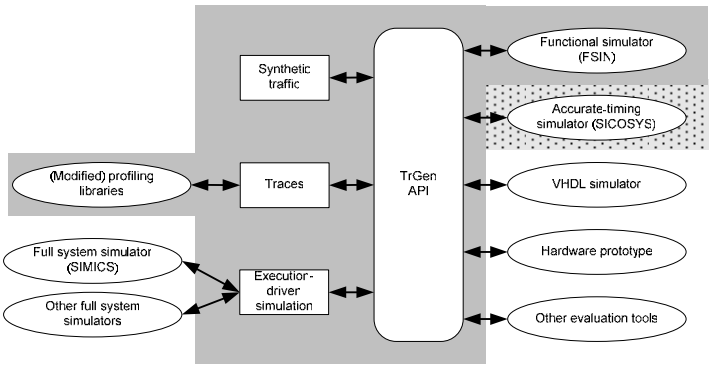

Figura 1. Diseño de INSEE. Una API común permite la conexión de diferentes simuladores con las fuentes de tráfico posibles.

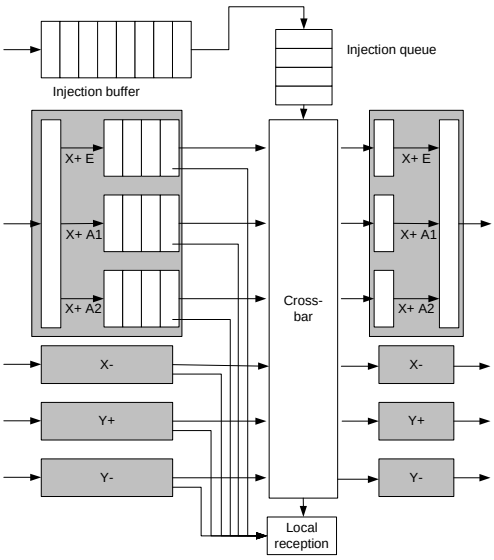

Figura 2. Modelo de encaminador simulado por FSIN.

2.2. TrGen

TrGen es una herramienta de generación de tráfico que, por medio de una API común permite probar los diseños de propuestas de redes de interconexión con diferentes fuentes de tráfico, como son tráfico sintético, tráfico tomado a partir de trazas reales y tráfico real sacado de una simulación conducida por la ejecución.

Tráfico sintético

TrGen permite la especificación de muchos tipos de tráfico sintético diferentes que pueden ser válidos para representar (siempre de forma aproximada) el intercambio de mensajes generado en la ejecución de aplicaciones reales.

El tipo de tráfico sintético se caracteriza por tres tipos de distribuciones: temporal, espacial y tamaño del paquete.

La distribución temporal determina el intervalo entre generaciones de paquetes, y sus correlaciones. Entre sus valores posibles están las distribuciones Bernouilli, ráfagas constantes, Markov.

La distribución espacial determina los nodos destino de los paquetes, y sus valores posibles son las distribuciones uniforme, zipf, zona caliente, punto caliente, y varias de tipo constante como transpuesta, mariposa, barajado perfecto, o inversa.

La distribución de tamaño del paquete determina el tamaño de los paquetes que son enviados y sus posibles valores son las distribuciones constante, uniforme y polinomial.

Tráfico basado en trazas

El tráfico sintético proporciona una buena orientación sobre el potencial de una propuesta de diseño. Pero las medidas de rendimiento obtenidas pueden no ser del todo ciertas, ya que en las aplicaciones reales los intercambios de mensajes se hacen en momentos determinados y normalmente siguen una relación de causalidad, esto es, una aplicación muchas veces espera la llegada de un mensaje para a su vez mandar otro de respuesta.

TrGen incorpora la posibilidad de utilizar trazas tomadas de tráfico real de aplicaciones paralelas como fuente de tráfico. Para ello se han realizado unas ligeras modificaciones a MPICH [1], una implementación de MPI [3], para poder aumentar el grado de detalle de generación de trazas. Además, se ha implementado un pre-procesador que adapta las trazas a un formato propio de TrGen.

Los cambios realizados a MPICH consisten en hacer visibles como operaciones punto a punto las operaciones colectivas MPI. Sin esta modificación, MPICH guarda como una única traza las operaciones colectivas, aunque internamente las implementa como una serie de operaciones punto a punto.

En el pre-procesamiento realizado a las trazas obtenidas de MPICH se elimina toda la información relativa a tiempos de los eventos, manteniendo, sin embargo, la ordenación temporal de los mismos, para conservar la causalidad. Se hace esto porque las marcas de tiempo de las trazas dependen en gran parte de la velocidad de proceso de los nodos y de la velocidad de la red de interconexión que los conectaba en la máquina real usada para la obtención de las trazas. De esta manera cuando hacemos una simulación basada en trazas sometemos a la red a la máxima presión, convirtiendo la red en el cuello de botella de la ejecución, para saber cuál es su potencial de rendimiento.

Tráfico real

Por último, TrGen permite la simulación con tráfico real, generado por aplicaciones en ejecución (en un entorno simulado). Explicamos esto con detalle en la próxima sección.

3. Simulación conducida por la ejecución

Como acabamos de adelantar, la tercera fuente de tráfico posible con TrGen es tráfico real generado y consumido por aplicaciones reales. Para ello se utilizan simuladores de sistemas completos que simulan computadores en los que se ejecutan las aplicaciones paralelas de interés que generan el tráfico que se inyectará en el simulador de redes de interconexión.

La simulación es pues tan realista como detallada es la descripción de cada componente de la simulación, y el precio a pagar por realizar una simulación conducida por la ejecución es una enorme pérdida de rendimiento con respecto de un sistema real, debido a la simulación tanto de la red que interconecta los computadores como de los propios computadores que ejecutan las aplicaciones paralelas.

En este momento la simulación de los nodos de cómputo se realiza utilizando el producto SIMICS de Virtutech [2]. Es un simulador que permite la simulación de una amplia variedad de arquitecturas hardware sobre un PC de sobremesa, por ejemplo, permite simular una máquina Sparc de 4 procesadores con el sistema operativo Solaris (*guest*) sobre un PC Intel con el sistema operativo Linux (*host*).

Actualmente usamos SIMICS para simular los nodos de cómputo (*guest*) sobre máquinas PC Intel con Linux de sistema operativo. Los nodos de cómputo simulados son sistemas x86 con 64MB de memoria y Linux Red Hat 7.3 como sistema operativo. Cada uno de estos nodos de cómputo ejecuta la aplicación paralela de interés y genera mensajes que son enviados al resto de los nodos de cómputo usando TrGen como interfaz de comunicación con el simulador de redes de interconexión elegido, en nuestro caso FSIN.

Podemos tener varios *hosts* simulando cada uno varios nodos de cómputo, *guests*, para así construir una red tan grande como nos permitan nuestros recursos, eso sí teniendo como limitación el número de licencias de SIMICS de que dispongamos, requiriendo cada *host* de una.

SIMICS también incluye un mecanismo que actúa como una red Ethernet simulada que sirve para conectar y además sincronizar a todas las instancias de los nodos de cómputo, denominado SIMICS Central. En nuestra configuración, sólo utilizamos Central para sincronizar entre sí los nodos de cómputo, ejecutando la simulación de una manera determinista, ya que nuestros nodos de cómputo se comunican a través de la red de interconexión simulada, FSIN, mediante la API de TrGen.

De esta manera, los mensajes generados por un nodo de cómputo (*guest*) son interceptados y enviados a TrGen que a su vez los inyecta en FSIN. Asimismo, cuando un mensaje que circula por la red de interconexión llega a su destino, es inyectado mediante TrGen en el nodo de cómputo correspondiente. Este intercambio de mensajes puede verse en la Figura 3.

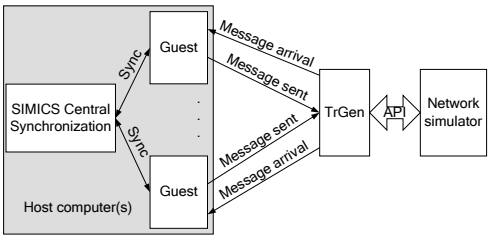

Figura 3. Interacciones entre los elementos involucrados en la simulación conducida por la ejecución.

3.1. Diseño de la arquitectura

Durante el desarrollo de este trabajo se han realizado dos implementaciones distintas para conectar el simulador de sistema completo SIMICS con una red de interconexión vía TrGen.

Implementación basada en *driver* y módulo SIMICS hardware

Esta versión está compuesta fundamentalmente por dos componentes desarrollados en su totalidad por el grupo: un módulo SIMICS que realiza las funciones de un adaptador de red PCI, y un *driver* para instalar en el núcleo de Linux que sirva como interfaz entre el sistema operativo del nodo de

cómputo simulado y el módulo de hardware desarrollado.

SIMICS permite extender su funcionalidad mediante la implementación de módulos software adicionales que se incorporan al simulador y que pueden realizar cualquier función, como por ejemplo, implementar la funcionalidad de una tarjeta de red PCI, que es nuestro caso.

Un nodo de cómputo ve este módulo como un adaptador de red PCI, al que se le instala un *driver* creado específicamente para él.

A las aplicaciones paralelas que se ejecutan en el nodo de cómputo se les indica que intercambien sus mensajes a través de la interfaz de red creada por nosotros y que se identifica en el sistema operativo como si de una tarjeta de red PCI se tratase. Dichos mensajes a inyectar en la red de conexión son recibidos por el *driver* de nuestra tarjeta de red, el cual escribe en los registros de nuestro adaptador de red, y que se encarga de sacarlos al exterior del simulador mediante una conexión TCP/IP e interactuar con TrGen para que éste inyecte los paquetes en el simulador de redes de interconexión.

Cuando un paquete de la red de interconexión llega a un nodo, es TrGen quien se comunica por TCP/IP con nuestro módulo hardware de SIMICS, quien provoca una interrupción hardware anunciando la llegada de un nuevo paquete, la cual es atendida por el *driver* de dicha interfaz de red, que a su vez se encarga de hacerlo llegar al sistema operativo y éste a la aplicación paralela. Toda esta comunicación puede verse en la Figura 4.

Por otra parte los nodos de cómputo se sincronizan entre sí como ya hemos comentado mediante el uso del módulo de SIMICS, Central, realizando así una ejecución determinista.

Esta implementación es válida para simular redes de interconexión conectadas con los nodos de cómputo por medio de adaptadores de red diferentes de Ethernet, ya que somos nosotros los que decidimos la implementación hardware del adaptador, y podemos hacer que se identifique en el *kernel* como un dispositivo de red no Ethernet, algo que no ha sido implementado. Por ejemplo, también sería posible la implementación hardware de un módulo SIMICS análogo al existente pero que se comportase como un adaptador de red tipo Myrinet. En este caso habría que modificar también el *driver* para que se identificase en el *kernel* como un dispositivo de ese tipo.

Implementación basada en llamadas a rutinas del usuario (*callbacks*)

En las versiones recientes de SIMICS se han implementado una serie de rutinas en la API de desarrollo que permiten el establecimiento de llamadas a rutinas definidas por el usuario ante un evento (*callback*). En particular, estos eventos pueden ser de transmisión o recepción de paquetes asociados a un adaptador de red.

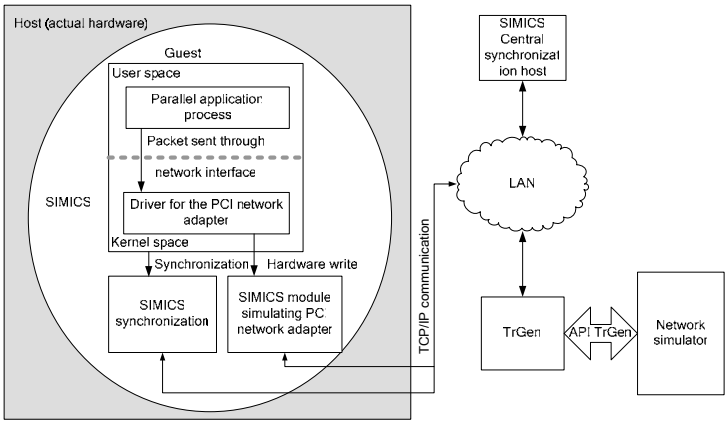

Figura 4. Estructura del intercambio de mensajes, desde su creación en una aplicación paralela (ejecutándose en un nodo de cómputo simulado) hasta su llegada al computador que ejecuta el simulador de redes de interconexión.

Con el uso de estos *callbacks* se simplifica mucho la creación del prototipo de simulación conducida por la ejecución, ya que se elimina la creación del módulo SIMICS que implementa el adaptador de red PCI, y su consiguiente *driver*.

Otra novedad en SIMICS es la posibilidad de manipular las interfaces de red, inyectando o eliminado paquetes de la misma.

En esta implementación la aplicación paralela ejecutada en los nodos de cómputo se comunica utilizando un adaptador de red de tipo Ethernet propio de SIMICS. Nosotros nos encargamos de crear un módulo de propósito general para SIMICS, porque no implementa ningún dispositivo hardware. En este módulo instalamos *callbacks* asociados al envío de paquetes por el adaptador de red que SIMICS nos proporciona.

Cuando una aplicación envíe un paquete, SIMICS se encargará de ejecutar el *callback* que hemos instalado, avisándonos de que tal envío se produce. Es entonces cuando utilizando las nuevas funcionalidades de SIMICS, en cuanto a dispositivos de red se refiere: le "quitamos" el paquete a la interfaz de red, lo sacamos fuera del entorno de simulación de SIMICS y se lo mandamos por TCP/IP a TrGen, que se encargará, al igual que antes de inyectarlo en nuestro simulador de redes de interconexión.

De manera análoga, cuando un paquete de la red de interconexión llega a su destino, TrGen se comunica con nuestro módulo SIMICS avisándole de la entrega del paquete. Entonces aprovechamos la funcionalidad de SIMICS para inyectar el paquete en la interfaz de red. Después de esto, SIMICS hará llegar al *driver* el paquete, y éste a su vez se lo transmitirá al sistema operativo y finalmente el paquete llegará a la aplicación paralela.

También es necesario implementar un módulo SIMICS de propósito general que se ejecutará dentro del proceso Central, y que será el encargado de instalar otro *callback*, esta vez para que cada cierto tiempo, predefinido, al que llamamos "paso" (de sincronización), mande un mensaje con el *timestamp* de la ejecución al proceso encargado de simular la red de interconexión, y se pare esperando a que dicho proceso le responda, permitiendo así la continuación de la simulación. De esta manera la simulación avanza de manera determinista, paso a paso, siendo la duración de este paso configurable desde el módulo SIMICS de los nodos de cómputo, como se comentará en el siguiente apartado.

Esta versión tiene, frente a la basada en la implementación de módulos hardware, la ventaja de que es más sencilla, porque se aprovechan las nuevas funcionalidades de SIMICS y toda la gestión de las comunicaciones es más simple, siendo SIMICS el que se encarga de hacernos llegar el paquete.

3.2. Esquema de comunicaciones

Los nodos de cómputo basados en SIMICS se comunican entre sí siguiendo el diagrama que puede observarse en la Figura 5. Por una parte se mantienen sincronizados entre sí conectándose al módulo de SIMICS Central. Como ya se ha comentado, Central sirve de infraestructura de red para comunicar los nodos de cómputo, (funcionalidad que no usamos) y de medio de sincronización entre los mismos.

Por otro lado, los nodos de cómputo envían sus paquetes fuera del entorno SIMICS, como se ha explicado en el apartado anterior, o bien a TrGen que los inyectará en el simulador de redes de interconexión, o a una aplicación de pruebas llamada *Switch* que actúa como un conmutador enviando los paquetes que le llegan de un nodo de cómputo a su destinatario. Esta aplicación está dotada de la capacidad de simular un retardo en las comunicaciones que denominamos retardo de interconexión, controlado por un parámetro definido por el usuario.

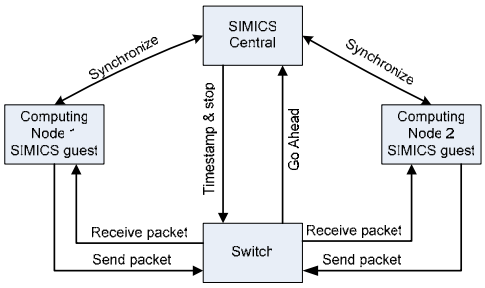

Figura 5. Comunicación establecida entre los distintos componentes del prototipo.

Como se ha comentado en el punto anterior, el módulo implementado dentro de Central, manda un mensaje al proceso *Switch* a cada paso, e interrumpe después su ejecución—lo que resulta

en una interrupción de la simulación, ya que los nodos de cómputo están sincronizados mediante Central.

Durante la ejecución normal entre dos mensajes de *timestamp* (un paso), los nodos de cómputo realizan sus tareas habituales, enviando sus mensajes al proceso *Switch* que los almacena hasta la llegada de un nuevo *timestamp*. La duración del paso la define el usuario.

El mensaje con el *timestamp* que llega al proceso *Switch*, le indica a éste que es el momento de realizar un envío de los mensajes que guarda encolados, hacia los nodos de cómputo de destino. Entonces el *Switch* se comunica con los módulos SIMICS que implementan la recepción y envío de comunicaciones (en los nodos) y les manda los mensajes que tiene pendientes; eso sí, indicándoles que no deben ser entregados hasta que pase un tiempo indicado como parámetro del proceso *Switch*, que emula el retardo de la red de interconexión.

Una vez la cola de mensajes pendientes de entrega en el *Switch* está vacía, éste manda el mensaje de *Go Ahead* a nuestro módulo SIMICS ubicado en el proceso Central, indicándole que puede continuar con la simulación durante un nuevo paso.

La ejecución de un paso en SIMICS supone, entre otras cosas, que los mensajes almacenados en las colas de los nodos sean inyectados en las interfaces de red de esos nodos en los momentos indicados por los eventos programados.

Todo este mecanismo puede ser visto gráficamente en la Figura 6.

3.3. Resultados

Se han realizado pruebas de funcionamiento del prototipo de simulación conducida por la ejecución. Las pruebas han sido satisfactorias debido a que el sistema se ha comportado como se esperaba, no se pierde ningún mensaje de los nodos de cómputo.

Las pruebas tipo han sido realizadas en un único SIMICS *host*, ya que no se disponía más que de una licencia de este simulador. En una misma máquina se han simulado 4 nodos de cómputo basados en Intel x86 con 64 MB de RAM y Red Hat Linux instalado. Estos nodos se conectan a la red por medio de una interfaz de red PCI Ethernet DEC 21143 de 100Mbs, proporcionada por SIMICS. Dichos nodos se comunican entre sí mediante el proceso *Switch* y se sincronizan utilizando Central.

Con esta configuración, se han calculado los siguientes valores como apropiados. Un ciclo SIMICS son 50 ns, pudiendo ser configurable, ya que este valor viene dado por la velocidad de reloj que se le indica al simulador en su configuración y que es de 20MHz por defecto. Se ha calculado que, debido al retardo introducido por la atención del sistema operativo a las interrupciones, y a que la interfaz de red de estas pruebas es una Ethernet, el tiempo mínimo entre envíos de paquetes es de unos 2200 ciclos, o sea, unos 110 µs. Con estos valores se ha configurado la latencia de sincronizaciones entre los nodos y Central para que sea de 50 µs, y el paso sea de 2000 ciclos, o 100 µs.

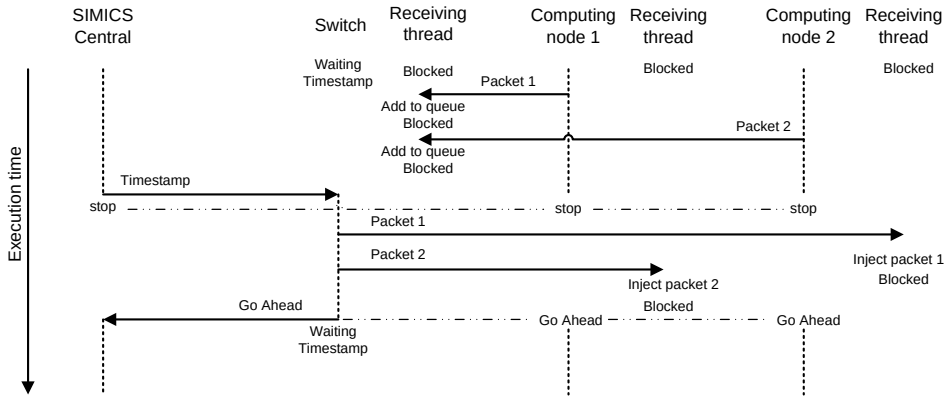

Figura 6. Diagrama de flujo de un intercambio de mensajes realizado entre dos nodos de cómputo.

De esta manera, los paquetes enviados por los nodos llegarán en el siguiente paso a sus destinos respectivos y se les indicará que sean inyectados en el sistema operativo una vez transcurrido el retardo de interconexión.

Se ha realizado un sencillo experimento, consistente en ejecutar una aplicación MPI (cpi, cálculo distribuido del valor de pi) en las 4 máquinas simuladas. La interconexión se produce mediante los adaptadores Ethernet y nuestro *Switch*. Variando el retardo de interconexión se han obtenido los datos recogidos en la Figura 7, en la que se puede observar que el tiempo de ejecución de programa paralelo es directamente proporcional al retardo de la red.

4. Conclusiones y trabajo futuro

En este trabajo hemos presentado las líneas generales de diseño e implementación de un sistema de simulación de redes de interconexión con fuentes de tráfico reales. Las ventajas de esta aproximación son evidentes, porque eliminan las dudas sobre la fidelidad de las fuentes de tráfico empleadas en la evaluación de una red. Los inconvenientes también quedan a la vista: el enorme coste asociado, que a efectos prácticos impide la simulación de redes de gran tamaño.

El sistema está en fase de pruebas; aún no se han hecho experimentos con simulaciones reales. Esta es, precisamente, la línea de trabajo actual. La experiencia que se gane al integrar el sistema con simuladores con FSIN conllevará, sin duda, mejorar su diseño y aumentar su funcionalidad.

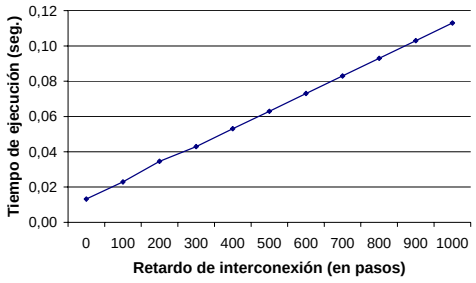

Figura 7. Tiempos de ejecución de cpi para distintos valores de retardo de interconexión.

Referencias

[1] W. Gropp, E. Lusk, N. Doss, A. Skjellum (1996). A high-performance, portable implementation of the MPI message passing interface standard, in Parallel Computing, vol. 22, no. 6.

[2] Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hållberg, Johan Högberg, Fredrik Larsson, Andreas Moestedt, Bengt Werner, (2002). Simics: A Full System Simulation Platform, IEEE Computer, February.

[3] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. Available at http://www-unix.mcs.anl.gov/mpi/standard.html.

[4] V. Puente, J.A. Gregorio, R.Beivide (2002). SICOSYS: An Integrated Framework for studying Interconnection Network in Multiprocessor Systems, Proceedings of the IEEE 10th Euromicro Workshop on Parallel and Distributed Processing. Gran Canaria, Spain.

[5] F.J. Ridruejo, A. Gonzalez, J. Miguel-Alonso. "TrGen: a Traffic Generation System for Interconnection Network Simulators". 1st. Int. Workshop on Performance Evaluation of Networks for Parallel, Cluster and Grid Computing Systems (PEN-PCGCS'05) (Olso, Norway, June 14, 2005).

[6] F.J. Ridruejo, J. Miguel-Alonso. "INSEE: an Interconnection Network Simulation and Evaluation Environment". Lecture Notes in Computer Science. (2005), to appear.