TrGen: sistema de generación de tráfico para simuladores de redes de interconexión

Francisco Javier Ridruejo Pérez, Antonio González Castro y José Miguel-Alonso

Resumen—**Presentamos un entorno de generación de tráfico capaz de interactuar con simuladores de redes de interconexión. Este entorno puede generar tráfico totalmente sintético, tráfico a partir de trazas de ejecuciones previas, y tráfico real obtenido de un simulador de sistemas.**

Palabras clave—**redes de interconexión, simulación, patrones de tráfico, aplicaciones paralelas.**

I. INTRODUCCIÓN

LAS redes de interconexión son una materia de gran interés en el ámbito del diseño de computadores paralelos y distribuidos.

En las etapas iniciales de diseño y evaluación de propuestas arquitectónicas para redes de interconexión (algoritmos de encaminamiento, estrategias de evitación de interbloqueos, mecanismos de tolerancia a fallos, topologías, etc.) es fundamental trabajar con simuladores funcionales que, aunque no incorporen todos los detalles de un sistema real, sí tengan en cuenta los más relevantes, de tal manera que se pueda, en un tiempo reducido, comprobar la viabilidad de la propuesta, antes de portarla a un simulador más detallado (como SICOSYS [7]) o de implementarla en hardware.

Tan importante como describir bien el sistema de interconexión en el simulador, es tener bien caracterizada la carga de trabajo a la que será sometido. Al igual que el diseño de los procesadores se realiza teniendo en consideración las cargas de trabajo que los van a utilizar, los sistemas de comunicaciones tienen que ser diseñados teniendo en cuenta cómo van a ser usados por las aplicaciones paralelas y distribuidas que se ejecutarán sobre ellos. No es lo mismo:

- Una red de interconexión para un SMP de pequeño tamaño
- Una red para una máquina CC-NUMA de gran tamaño
- Una red para un MPP de miles de nodos
- Una red para aplicaciones distribuidas basadas en servicios Web

De hecho, el mercado ha respondido con soluciones diferentes a cada una de estas necesidades. Pero podemos ir más allá: tampoco es igual lo que exigimos de la red de un MPP cuando ejecutamos una aplicación tipo maestro-esclavo (donde las interacciones son poco frecuentes, pero con un intercambio de mensajes largos) que cuando ejecutamos un simulador paralelo (en el que se intercambian frecuentemente mensajes muy cortos) [5]. En el primer caso nos interesa un throughput muy elevado, en términos de MB/s. En el segundo, queremos que la latencia sea mínima, para que la red no se convierta en un cuello de botella.

Por todo lo dicho, en un sistema de simulación-evaluación de propuestas para redes de interconexión, un elemento fundamental es el método de generación de tráfico. Se pueden usar muchas técnicas:

1. Patrones de tráfico sintéticos: tráfico uniforme, matriz transpuesta, hot-spot... Son muy sencillos de implementar en un simulador, y en muchos casos están diseñados para emular el comportamiento de aplicaciones de cálculo científico. Se puede criticar (de hecho, se hace con frecuencia) lo poco representativos que resultan estos patrones, pero suministran información importantísima en las fases preliminares del diseño.
2. Trazas de aplicaciones reales: se ejecuta la aplicación en un sistema mono o multi-procesador, pero usando múltiples procesos que se comunican, y se usan las trazas obtenidas para alimentar el simulador de una red diferente. Esta técnica está limitada por el tamaño del sistema empleado para obtener las trazas.
3. Simulación conducida por la ejecución: se simula un sistema completo, tanto el subsistema de comunicación como la colección de nodos de cómputo. Se ejecuta sobre este sistema la aplicación objeto de nuestro estudio. Evidentemente, así se obtienen los datos más realistas. El problema es lo lenta que puede resultar una simulación hecha de esta manera.

Por otra parte, ¿qué aplicación es representativa? Si bien es cierto que se construyen computadores paralelos con una única aplicación en mente (o un conjunto muy reducido de ellas), lo cierto es que comercialmente interesan los computadores de propósito más general, en los que no se ejecutan aplicaciones de tipo “científico”, sino de tipo “comercial”, entre los cuales podríamos destacar los sistemas basados en transacciones sobre grandes bases de datos. Debe señalarse que el comportamiento de esta clase de cargas de trabajo difiere considerablemente del de las cargas de trabajo de tipo numérico desde el punto de vista de la presión ejercida sobre el sistema computador [10].

El objetivo de este trabajo es presentar **TrGen**, un entorno completo de generación de tráfico para redes de interconexión, de tal forma que los investigadores que usan simuladores puedan centrarse en el modelado de sus redes, y utilizar TrGen para alimentarla.

Departamento de Arquitectura y Tecnología de Computadores. Universidad del País Vasco. Apartado 649, 20080 San Sebastián. E-mail de contacto: miguel@si.ehu.es.

El resto del artículo está organizado de la siguiente manera. En la Sección II describiremos el modelo de funcionamiento de TrGen. Después, daremos algunos detalles de la implementación de la herramienta en las secciones III (tráfico sintético y trazas) y IV (tráfico conducido por ejecución). Terminamos con una serie de conclusiones, así como una enumeración de las líneas de trabajo futuro, en la Sección V.

II. Diseño del generador de tráfico TrGen

A. Diseño

El diseño de TrGen parte del esquema representado en la Fig. 1. A la derecha tenemos diferentes entornos de simulación para redes. A la izquierda, diferentes patrones de tráfico: totalmente sintéticos, semi-sintéticos, trazas, o generados por las aplicaciones. Los patrones semi-sintéticos serían aquellos que, generando el gráfico de forma programática, están parametrizados a partir de datos obtenidos de trazas o de ejecuciones reales.

Sea cual sea la fuente de tráfico, los simuladores disponen de un API único para acceder a ella. Este API consta de tres funciones básicas:

1. Inicialización: selección del patrón a emplear, y parametrización del mismo. La parametrización será muy distinta en cada caso: puede tratarse de un fichero a abrir, de la media de una función de generación de números aleatorios, o la dirección IP de un servidor de tráfico conectado a un sistema SIMICS [9].
2. Solicitud de mensajes a inyectar en la red.
3. Notificación de mensajes recibidos de la red. Esto es fundamental cuando estamos conectados a una aplicación en ejecución, o cuando el tráfico es reactivo (la generación de nuevos paquetes está condicionada a la recepción de paquetes anteriores).

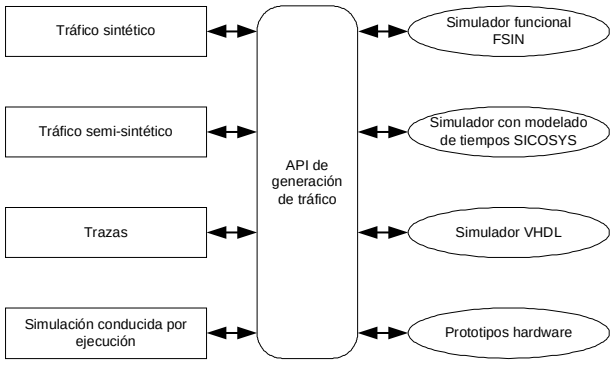

Fig. 1. Modelo de funcionamiento de TrGen. A través de una interfaz única, los simuladores de redes de interconexión acceden a muy diversas formas de generadores de tráfico.

B. API

En estos momentos disponemos de un prototipo de TrGen que implementa un API básico, sujeto a futuras ampliaciones. Dicho API consta de las siguientes funciones:

```
void source_new(source_t * s);
```

```
void source_init(source_t s, long clock,
source_e type, ...);
```

Estas funciones permiten crear una fuente de tráfico e inicializarla. Los parámetros relevantes son *s*, la fuente de tráfico, y *type*, que define el tipo de tráfico que generara la fuente. En estos momentos, las clases de tráfico disponible son STREAMED para tráfico sintético y FILED para trazas de aplicaciones reales. La función `source_init()` recibe un número de parámetros variable que depende del tipo de tráfico.

```
t_packet * source_next(source_t s, long *
npackets);
```

Devuelve el vector de paquetes y el número de paquetes generado en el ciclo actual por la fuente *s*.

```
void source_notify(source_t s, t_packet
packet);
```

Esta función permite a la red notificar a la fuente de tráfico la entrega de un paquete generado anteriormente por la fuente. Esta función sólo tiene relevancia cuando hay relaciones de causalidad entre los paquetes, como por ejemplo en los tráficos generados a partir de trazas o en las simulaciones conducidas por la ejecución.

```
bool_t source_finished(source_t s);
```

Esta función se emplea para averiguar si la fuente de tráfico tiene o no más paquetes para entregar. La condición de finalización depende del tipo de tráfico generado, pero puede tener en cuenta criterios como el máximo número de ciclos o de paquetes entregados, o el final del fichero de trazas.

De momento, esta API está indicada para simuladores conducidos por el tiempo. En el futuro, se ampliará para adaptarse a simuladores conducidos por eventos.

III. Tráfico sintético y basado en trazas

A. Tráfico sintético

El tráfico sintético es el más sencillo de generar. Está caracterizado por tres distribuciones: temporal, espacial y de tamaño del paquete.

1. La distribución temporal determina el tiempo de llegada entre paquetes, y su correlación.
2. La distribución espacial determina los nodos de destino de los paquetes.
3. La distribución del tamaño determina el tamaño de los paquetes generados.

A modo de ejemplo, una forma frecuente de evaluar en la fase preliminar una red es usando tráfico sintético con una distribución temporal de Bernouilli, una distribución espacial uniforme o con un patrón muy regular, y un tamaño de paquete constante. Es posible sin embargo especificar fuentes de tráfico sintético muy diversas, para simular distintos tipos de aplicaciones. Las posibilidades para cada distribución son muy amplias, permitiendo cada una, según su tipo, evaluar distintos aspectos de la red.

TrGen define el concepto de flujo para definir una fuente de tráfico sintético. Cada flujo tiene un origen, una distribución espacial, una distribución temporal y una distribución de tamaño. Cada origen puede tener varios flujos distintos, permitiendo modelar el comportamiento del tráfico con mayor precisión. Las distribuciones son parametrizables, y la colección de distribuciones disponibles es extensible. En el prototipo actual están disponibles las siguientes:

- Temporales:
  - Bernouilli
  - Ráfagas constantes
  - Markov (con distintas variantes)
- Espaciales:
  - Uniforme
  - Distribuida
  - Zipf
  - Hotspot
  - Constante: transpuesta, mariposa, barajado, inversa, complementaria, literal, etc.
- De tamaño del paquete:
  - Constante
  - Uniforme
  - Polinomial

Estas distribuciones son frecuentes en la literatura. Para una descripción más detallada, ver [2,3].

B. Trazas

Aunque es sencillo hacerse una idea de la carga que es capaz de soportar la red con patrones de tráfico sintético, los resultados obtenidos pueden ser poco realistas: las aplicaciones no muestran, normalmente, patrones de comunicación aleatorios, por lo que es difícil modelarlas con tales patrones. Los procesos se comunican entre sí en un orden determinado y momento preciso. Además, el envío de un mensaje suele producirse como consecuencia directa de la recepción de otro. El patrón de comunicación viene definido, pues, por una relación de causalidad entre los envíos y recepciones de mensajes entre los nodos. Con TrGen es posible emplear trazas de ejecución de aplicaciones reales para generar tráficos que sigan con fidelidad los patrones definidos explícitamente en las mismas, al menos en lo que se refiere a la distribución espacial, a las relaciones causales, y a los tamaños de los paquetes.

Para obtener ficheros de trazas con las que alimentar a TrGen empleamos una versión modificada de MPICH, una de las implementaciones más populares de MPI [4]. MPICH incorpora un mecanismo sencillo de obtención de trazas de aplicaciones en ejecución, aunque estas trazas no son útiles para nuestros propósitos: las operaciones colectivas aparecen como tales en el fichero de trazas, sin reflejar el intercambio real de mensajes que, en una red, representa dicha operación (suponiendo redes *sin* soporte nativo para operaciones colectivas). Internamente, MPICH implementa las operaciones colectivas sobre operaciones punto a punto cuando no hay alternativas mejores. Nuestra modificación ha consistido en hacer visibles esas operaciones, y registrarlas en el fichero de trazas en lugar de la operación colectiva correspondiente.

Los ficheros de trazas necesitan cierto preproceso para poder ser empleados con TrGen. Este preproceso consiste en una selección de los campos relevantes (tipo de evento, nodo en que ocurrió, nodo que lo provocó y tamaño del evento) y un cambio de formato. Sería posible emplear otros métodos para obtener ficheros de trazas utilizables por TrGen, por ejemplo, tomando como punto de partida las capturas de tráfico obtenidas por un *sniffer* en una red Ethernet, y aplicando el preproceso oportuno.

Parte del preproceso realizado con las trazas es la eliminación de los *timestamps* de los eventos que las forman, manteniendo el orden temporal y las relaciones causales. Así sometemos a la red simulada a la máxima "presión" posible, como si fuese utilizada por unos procesadores infinitamente rápidos, y comprobamos de esta manera hasta qué punto la red es un cuello de botella para la ejecución de la aplicación. Una futura ampliación de TrGen consistirá en tener en cuenta la velocidad de procesamiento de los nodos, escalando la separación temporal de los eventos en función de esta velocidad.

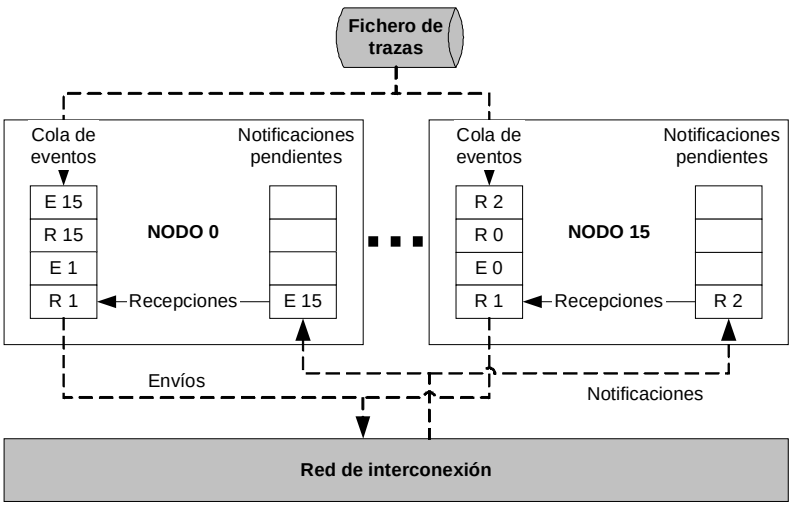

Fig. 2. Estructuras de datos utilizadas para mantener la causalidad en el tráfico basado en trazas. El nodo 0 está a la espera de la recepción de un paquete del nodo 1, al que luego responderá. Después esperará a recibir un paquete del nodo 15, paquete que ya ha sido recibido, puesto que se encuentra en la lista de notificaciones pendientes.

La figura 2 nos permite hacernos una idea de cómo funciona el tráfico basado en trazas. Cada nodo de la aplicación simulada tiene una cola de eventos. La cola se alimenta a partir del fichero de trazas e inyecta un paquete en la red cuado en su cabecera hay un envío. Si en la cabecera hay una recepción, comprobará si dicha recepción ha sido ya notificada y, si no es así, esperará a que la red la notifique. Cada nodo tiene una lista de notificaciones pendientes. Estas notificaciones se corresponden a eventos de recepción que aún no deben ocurrir y se guardan en la lista hasta que llegue su turno. De esta forma se retiene la causalidad de la traza.

IV. TRÁFICO CONDUCIDO POR LA EJECUCIÓN DE APLICACIONES

A. *Esquema general*

Como ya se ha comentado, para realizar una simulación conducida por la ejecución hace falta simular cada uno de los nodos de cómputo y la red de interconexión que los conecta. Sobre este sistema simulado se ejecutará luego la aplicación de interés, resultando así una simulación tan realista como lo sea la simulación de los componentes del sistema. Eso sí, la ejecución será varios órdenes de magnitud más lenta que en un sistema real, debido a la simulación mediante software tanto de los nodos de cómputo como de la ejecución de aplicaciones paralelas en esos nodos simulados.

Para la simulación de los nodos de cómputo en los que ejecutamos las aplicaciones paralelas de interés se utiliza SIMICS de Virtutech [9], que permite la simulación con todo nivel de detalle de sistemas completos sobre un amplio rango de plataformas hardware aún sin disponer de las mismas. Por ejemplo, permite simular una Sparc de 4 procesadores, con Solaris de sistema operativo y varios GB de RAM (sistema huésped) sobre un PC con procesador Intel y con el sistema operativo Windows (sistema anfitrión).

Para simular la red de interconexión que permite a los nodos de cómputo comunicarse se utiliza el simulador de redes de interconexión de nuestra elección, través de la interfaz TrGen.

En nuestras pruebas, estamos ejecutando SIMICS bajo Linux en PCs convencionales para simular nodos de cómputo basados en procesadores x86 con 256 MB de RAM. Cada uno de estos nodos de cómputo ejecuta la aplicación paralela que genera mensajes para los demás nodos. Podemos tener varios PC anfitriones, cada uno de ellos simulando varios PC huéspedes, para llegar a una red simulada del tamaño deseado (limitado por nuestros recursos).

SIMICS incorpora un mecanismo denominado SIMICS Central para sincronizar todos los equipos simulados. También actúa como LAN simulada, de tal forma que los PCs huéspedes pueden comunicarse. Nosotros mantenemos la función de sincronización de SIMICS Central, pero reemplazamos la de comunicación por TrGen más nuestro simulador de redes: los mensajes generados por un PC huésped son interceptados y enviados a TrGen, que se los pasará al simulador de redes. Del mismo modo, los mensajes generados por el simulador serán inyectados desde TrGen al PC huésped indicado. Todo esto puede verse en la Figura 3.

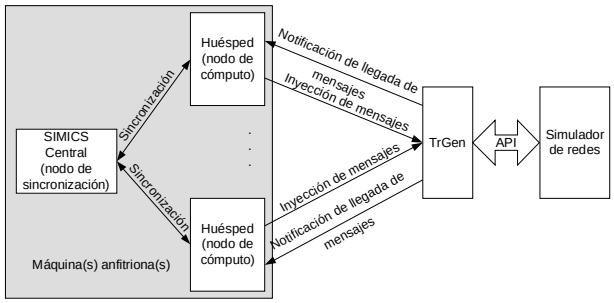

Fig. 3. Esquema de conexión entre los nodos de cómputo (huéspedes, simulados por una instancia de SIMICS), el coordinador SIMICS Central, TrGen y el simulador de la red de interconexión.

Ya hemos dicho que los huéspedes o nodos de cómputo simulados son máquinas virtuales SIMICS que se ejecutan como un proceso cualquiera dentro del host. En principio, estos nodos sólo pueden comunicarse entre sí y salir a la red local a través de la conexión con los servicios que proporciona el módulo SIMICS Central, entre los que están ARP, BOOTP y DHCP. El más importante, desde nuestro punto de vista, es la sincronización entre las distintas máquinas virtuales SIMICS, para que el simulador en conjunto actúe de una manera totalmente determinista. Para conectar un nodo de SIMICS con la red local real del anfitrión que está ejecutando SIMICS, se puede utilizar SIMICS Central como si de un encaminador IP se tratase.

B. *Detalles de la implementación*

Para hacer que los mensajes generados por las aplicaciones paralelas, que son ejecutadas en los nodos de cómputo, lleguen a la red de interconexión simulada, y que dichos mensajes pasen de la red de interconexión simulada al nodo de cómputo que los tiene que recibir, se ha diseñado un complejo mecanismo que explicamos a continuación. No vale servirse de los servicios proporcionados por SIMICS Central, porque éste tiene una implementación cerrada e implementa servicios basados en una red Ethernet.

En cada anfitrión se ejecutan una o varias instancias de SIMICS, cada una simulando un huésped. Cada huésped es un PC con 256 MB de RAM, procesador Intel x86, dos adaptadores de red (uno Ethernet ISA y otro PCI) una tarjeta gráfica Voodoo3 de 16 MB PCI y un disco duro IDE. El sistema simulado tiene instalado el sistema operativo Linux RedHat 7.3. Evidentemente, todo el hardware de este huésped está simulado mediante software usando los módulos que provee SIMICS.

Una vez instalado, configurado y arrancado el sistema operativo de cada huésped, se instala la aplicación paralela a ejecutar, por ejemplo, uno cualquiera los NAS Parallel Benchmarks [6].

Si usamos, sin más, los servicios de SIMICS Central, tenemos una simulación de un pequeño cluster de PCs, con Ethernet (por supuesto, simulada) como red de interconexión. El paso siguiente es sustituir esa red por otra distinta. Para ello hemos introducido en SIMICS la simulación de un adaptador de red que siga la especificación PCI y que actué como puente con nuestra

red. Desde el punto de vista de la máquina huésped, se trata de un segundo adaptador Ethernet. Desde nuestro punto de vista, será la interfaz con TrGen. El primer adaptador Ethernet (ISA) se mantiene, conectado a SIMICS Central, para conservar la sincronización entre elementos.

Cada vez que la aplicación paralela ejecutándose en un huésped envía un mensaje MPI a otro proceso (residente en otro nodo de cómputo), lo hace a través de su adaptador de red (el PCI). El mensaje llega primero al driver de ese adaptador, que se encuentra en el espacio de núcleo del sistema simulado. Dicho driver ha sido creado por nosotros, basándonos en [1] y [8], y se encarga de leer/escribir en el adaptador de red PCI simulado, que está implementado como un módulo creado en C e incorporado al sistema SIMICS mediante su sistema de extensión y creación de nuevos módulos y dispositivos.

Los mensajes generados por la aplicación se envían al exterior mediante escrituras realizadas por el driver en los registros y en la memoria del adaptador PCI simulado. Tales escrituras en realidad son interceptadas por el módulo creado en SIMICS, que es en última instancia el punto de contacto con TrGen.

Del mismo modo, este módulo se encarga de recoger los mensajes que le llegan desde TrGen e inyectarlos en el huésped correspondiente mediante la escritura en su memoria de dispositivo PCI. También realizará notificaciones en forma de interrupciones. Ante una interrupción, el driver le da servicio y se encarga de mandar el mensaje recibido a la aplicación que lo está esperando.

Como se puede comprobar, las piezas clave en nuestro diseño son (1) el módulo SIMICS que simula un adaptador de red conectada a un bus PCI, y (2) el driver, para el kernel de Linux, que interactúa con ese dispositivo.

La comunicación entre los dispositivos PCI simulados y TrGen puede realizarse de múltiples formas (por ejemplo, mediante una serie de conexiones TCP). TrGen actúa como puente con el simulador de red de interconexión elegido, que será el que incorpore todos los mecanismos propios de una red de interconexión para un *cluster* o un multicomputador.

La Figura 4 representa todo el intercambio de mensajes (simulados y reales) implicados en una comunicación entre máquinas simuladas.

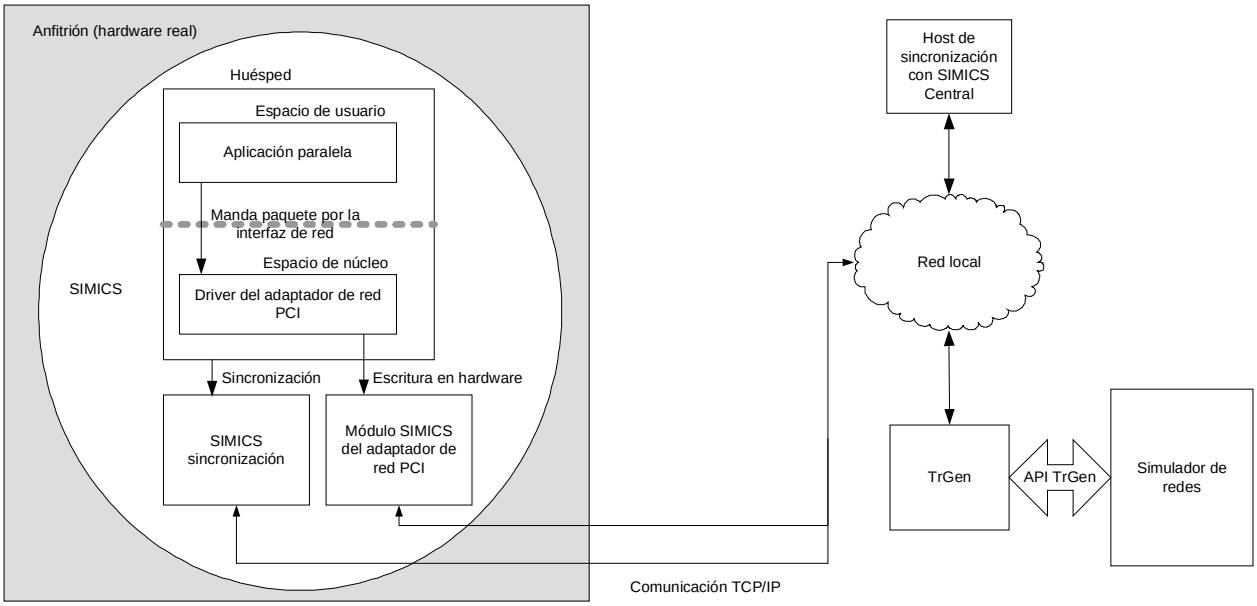

Fig. 4. Estructura del paso de los mensajes creados, desde su creación por la aplicación paralela, ejecutada en el nodo de cómputo simulado mediante SIMICS, hasta su llegada al host que simula la red de interconexión.

V. Conclusiones y líneas de trabajo abiertas

A. Estado de TrGen

TrGen es un proyecto vivo, que está aún en desarrollo. Como se ha ido avanzando a lo largo de este artículo, estamos considerando bastantes ampliaciones y mejoras. Por el momento, resulta funcional la parte encargada del tráfico sintético, con las distribuciones antes enumeradas. También está funcional el sistema de captura, adaptación y generación de tráfico basado en trazas. La generación de tráfico real, mediante SIMICS, está en pruebas.

Se han hecho tests de interoperabilidad con un simulador funcional desarrollado por nuestro grupo de investigación, con resultados satisfactorios: aumentamos considerablemente la gama de experimentos que podemos realizar, sin que por ello se dispare ni el consumo de memoria ni el uso de CPU (comparando con la alternativa de que el propio simulador genere su tráfico).

B. Líneas abiertas

Por supuesto, nuestros planes actuales se centran en completar lo que aún no está terminado de nuestro diseño, y en realizar pruebas exhaustivas de funcionamiento. Pero estas no son las únicas líneas abiertas.

La generación de tráfico abarca muchas posibilidades y TrGen aun esta en su infancia. Entre los posibles

trabajos futuros se encuentran la ampliación de tipo de distribuciones disponibles: distribuciones espaciales con memoria, inclusión de patrones de tráfico reactivo y definición de tipos de tráfico de aplicaciones concretas (por ejemplo, FTP, WWW, POP...) como modelos sintéticos, y distribuciones espaciales que tengan en cuenta la distancia entre nodos. También queremos mejorar el tratamiento de eventos asíncronos para el tráfico basado en trazas, e incorporar la generación de patrones de tráfico semi-sintético. Todo esto supondrá una ampliación del API.

Por otra parte queda pendiente la integración con simuladores más sofisticados que los probados hasta ahora (a corto plazo buscaremos la integración con SICOSYS), y la realización de un TrGen-Lite, o versión ultraligera de TrGen para entornos de simulación concretos en los que se necesita un consumo de recursos mínimo.

AGRADECIMIENTOS

Este trabajo ha sido realizado con el apoyo económico del Ministerio de Ciencia y Tecnología (TIC2001-0591-C02-02) y de la Diputación Foral de Gipuzkoa (OF-758/2003).

REFERENCIAS

[1] Donald Becker, Linux network drivers, dentro de la web corporative de Scyld en http://www.scyld.com

[2] Jose Duato, Sudhakar Yalamanchili, Lionel Ni. Interconnection Networks: An Engineering Approach. Morgan Kaufmann, 2002.

[3] Itamar Elhanany (Ed.) Fabric Benchmarking Traffic Models Rev. 1.0. Network Processing Forum, 2003. Available at http://www.npforum.org/techinfo/BM_Fabric_TrafficIA.pdf

[4] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. Disponible en http://www-unix.mcs.anl.gov/mpi/standard.html

[5] J. Miguel, A. Arruabarrena, R. Beivide y J.A. Gregorio. “Assessing the performance of the new IBM SP2 communication subsystem”. IEEE Parallel and Distributed Technology, Vol. 4, nº 4 (1996), 12—22.

[6] NASA. The NAS Parallel Benchmarks. http://www.nas.nasa.gov/Software/NPB.

[7] V. Puente, J.A. Gregorio, R.Beivide. 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. January 2002.

[8] Alessandro Rubini, Jonathan Corbet. Linux Device Drivers 2nd Edition. O’Reilly.

[9] Virtutech, Inc. Página web de Simics (http://www.virtutech.se/simics/simics.html), dentro de la web corporativa de Virtutech, Inc. (http://www.virtutech.se/)

[10] S. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. “The SPLASH-2 programs: Characterization and methodological considerations”. In Proceedings of the 22nd International Symposium on Computer Architecture, June 1995.