Herramienta de simulación basada en formalismo DEVS para predicción de performance de redes de computadoras

Iván Melgrati, Carlos Giorgetti, Ana Tymoschuk
Grupo de Investigación de Redes GIRED – Proyecto O25/046
Departamento de Sistemas
Universidad Tecnológica Nacional – Facultad Regional Santa Fe
Lavaise 610 –3000 – Santa Fe
imelgrat@frsf.utn.edu.ar, cgiorget@frsf.utn.edu.ar, anrotym@alpha.arcride.edu.ar

Resumen

El presente trabajo plantea un modelo de simulación para una red de computadoras especificado con el formalismo DEVS (Discrete Event System Specification), para predecir el desempeño durante el procesamiento de cargas de trabajo. La red considerada es un sistema cliente-servidor, donde cada recurso se representa como un modelo atómico DEVS, y el conjunto, como un modelo acoplado. Se simula mediante la heramienta CD++. Simultáneamente, se desarrolla una herramienta gráfica (denominada VISUAL DEVS) para facilitar la especificación y posterior refinamiento de los modelos anteriormente mencionados. A partir de la corrida de estos modelos, se obtienen medidas de performance del sistema, para distintos conjuntos de requerimientos de las estaciones clientes al/los servidor/es.

Palabras claves: modelos de simulación, redes de computadoras, formalismo DEVS, predicción de performance.

Introducción

Los constantes cambios tecnológicos en el área de la informática y las nuevas demandas y exigencias de los usuarios hacen tener muy en cuenta el comportamiento actual del sistema o performance a fin de implementar las modificaciones necesarias o apropiadas en software y hardware y satisfacer los requerimientos de los usuarios o de las nuevas aplicaciones. Dicha performance se analiza a través de los índices más importantes como la velocidad de procesamiento de requerimientos o transacciones (throughput), el tiempo de respuesta del sistema, la utilización de los recursos, la velocidad de transmisión de paquetes, las longitudes de colas de trabajos, entre otros.

De las herramientas conocidas para la evaluación de performance, la simulación permite obtener y predecir los índices de comportamientos en distintos y eventuales escenarios. El enfoque más adecuado para el planteo del modelo de simulación de las redes de computadoras es el de eventos discretos [5], por la característica aleatoria de las cargas de trabajo y de los tiempos de servicio de los recursos para procesar dichas cargas. Los cambios de estado del sistema están provocados por eventos de entradas y salidas, en este caso los requerimientos de los clientes al servidor y las respuestas correspondientes.

Cada uno de los recursos físicos del sistema se especifica mediante el formalismo DEVS, conformando un modelo atómico, en el cual se consideran transiciones de estado internas y externas, y las entradas y salidas del modelo. Siguiendo siempre el formalismo DEVS, los modelos atómicos pueden acoplarse de manera jerárquica, hasta completar la representación de todo el sistema. Los modelos acoplados pueden ser tratados, a su vez, como nuevos modelos atómicos, de tal manera que se puede modelar un sistema altamente complejo, utilizando una estrategia modular y jerárquica. Este paradigma además facilita la modificación de los modelos, reduciendo el tiempo

de desarrollo de los mismos, a la vez que permite la reutilización directa de modelos atómicos y/o acoplados.

Objetivo del trabajo

En el ambiente informático, comúnmente se presenta la necesidad de evaluar la performance de un sistema de redes de computadoras. En el presente trabajo se plantea un sistema muy común del área informática, el cliente-servidor (C/S), cuyo esquema puede verse en la figura 1. El conocimiento de su performance es de vital importancia a la hora de tomar decisiones respecto al hardware y al software que lo conforma, para garantizar un nivel de servicio adecuado.

Se propone para dicha evaluación un modelo de simulación utilizando el formalismo DEVS. En él se especifican los componentes más relevantes del sistema como modelos atómicos, para acoplarlos y formar el modelo final. La simulación se lleva a cabo mediante la herramienta CD++ [1], la cual permite la especificación de los modelos atómicos en lenguaje C++. Paralelamente, se desarrolla una herramienta gráfica para la especificación de los modelos acoplados, con el fin de acelerar el proceso de creación y refinamiento de los mismos.

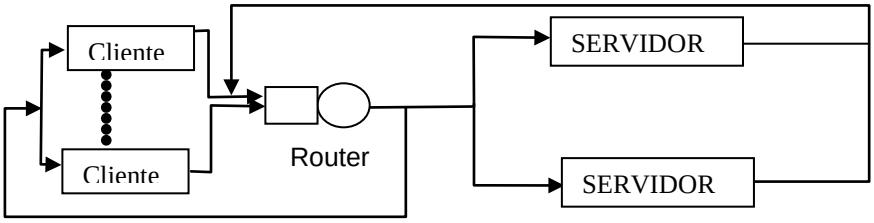


Figura 1: esquema del sistema cliente-servidor modelado

Los indicadores de la performance tenidos en cuenta en este artículo son: el tiempo de respuesta para una petición de un cliente, la velocidad de procesamiento del/los servidor/es, la cantidad de trabajos en espera o longitudes de cola en los distintos componentes del sistema. Se analiza el impacto en los indicadores de performance del modelo de los cambios en los parámetros siguientes:

- ✓ La modificación de cargas de trabajo arribando a los servidores.
- ✓ La variación de la cantidad de recursos disponibles.
- ✓ La modificación de las características de los recursos en el subsistema de comunicaciones (routers, redes, etc.)
- ✓ La modificación en el número de usuarios o clientes conectados al sistema.
- ✓ Los cambios en los parámetros de los servidores: tiempos de servicio de CPU, tiempos de servicio de discos, etc.

Formalismo DEVS: Especificación Formal

DEVS (**D**iscrete **Ev**ent **S**ystem Specification – Especificación de Sistemas de Eventos Discretos) [4] [6] [7] [8] es un formalismo universal para modelar y simular DEDS (**D**iscrete **E**vent **D**ynamic **S**ystems – Sistemas Dinámicos de Eventos Discretos), en los que las entradas, estados y salidas son constantes por intervalos; y cuyas transiciones se identifican como eventos discretos. Los intervalos de tiempo entre ocurrencias de eventos son variables, ofreciendo ciertas ventajas frente a los formalismos con una granularidad de tiempo única; donde es difícil describir los modelos debido a que hay muchos procesos operando en distintas escalas de tiempo. Esto hace que

los estados deban actualizarse a intervalos fijos, iguales al del modelo con menor tiempo entre ocurrencia de eventos consecutivos.

Un modelo DEVS se construye sobre la base de un conjunto de modelos básicos llamados **Atómico**s, que se combinan para formar modelos **Acoplados,** y un conjunto de relaciones que indican cómo los modelos están conectados entre sí, formando una estructura jerárquica. Los modelos atómicos se definen como una tupla consistente en:

$$\mathbf{M = < X, S, Y, \delta int, \delta ext, \lambda, ta >}$$

donde:

X: Conjunto de eventos externos, es decir el conjunto de eventos que pueden arribar a un modelo.

S: Conjunto de estados secuenciales. Es decir, es el conjunto de estados posibles que están definidos para el modelo.

Y: Conjunto de eventos externos generados como salida. Este conjunto define todos los eventos de salida (mensajes) que un modelo puede enviar.

δint: Función de transición interna. Esta función define la manera en la que ocurren los cambios internos del estado de un modelo. Es decir, aquellos cambios que se producen sin la necesidad de eventos externos.

δext: Función de transición externa. Esta función define cómo el modelo reacciona ante la ocurrencia (arribo) de un evento (mensaje) proveniente del exterior (otro modelo o una entrada externa al sistema).

λ: Función de salida. Esta función define cuáles son los mensajes a enviar a otros modelos. Estos mensajes (eventos) dependen del estado actual del modelo.

ta: Función de avance de tiempo (en inglés: time advance). Esta función define cuánto tiempo debe transcurrir para que el siguiente cambio de estado interno ocurra, si no ocurre ningún evento externo. Este valor depende del estado actual del modelo.

Los modelos básicos pueden acoplarse para componer un modelo, denominado modelo acoplado, definido como:

$$\mathbf{DN=<D, \{Mi\}, \{Ii\}, \{Zi,j\}, Select>}$$

donde:

D: Conjunto de nombres de los componentes. Estos identifican a los componentes atómicos que componen el modelo acoplado.

Mi: Modelo básico (atómico) correspondiente al componente i.

Ii: Conjunto de influencias del componente i. Es decir, el conjunto de modelos a los cuales puede influenciar (enviar mensajes) el modelo i.

Zi,j: Función de traducción de la influencia i. Es una función de mapeo entre los puertos de entrada de un modelo i y los de entrada de otro modelo j. Es decir, especifica como los modelos se “acoplan”.

Select: Función utilizada para determinar prioridades de ejecución de modelos. Esta función se utiliza para determinar el orden de ejecución de los modelos, en caso de que 2 ó más modelos deban ejecutar alguna función de transición al mismo tiempo.

Mediante el uso de transformaciones matemáticas es posible convertir a los modelos DEVS acoplados en Modelos DEVS atómicos, los cuales se pueden utilizar para formar otros modelos acoplados. Esta capacidad permite al modelador construir modelos muy complejos utilizando una estrategia modular, a través del modelado jerárquico. Otra ventaja de este formalismo es la capacidad de validar cada modelo (atómico o acoplado), en lugar de tener que realizar las pruebas sobre el modelo completo.

Descripción del modelo de la red de computadoras

En los sistemas cliente-servidor, uno o más clientes y uno o más servidores, junto con el sistema operativo subyacente, el sistema de comunicación entre procesos y los protocolos de la red forman un sistema compuesto que permite el procesamiento distribuido.

En la figura 2 se representa esquemáticamente el sistema, detallando la composición del servidor 1 con sus recursos físicos, representados mediante círculos y las colas de trabajos o solicitudes representadas mediante rectángulos.

El usuario, a través de una estación cliente, genera demandas al/los servidor/es que se transmiten mediante un mecanismo de comunicación entre procesos en los nodos del sistema. El/los servidor/es proporciona/n servicios a las estaciones cliente en respuesta a sus demandas. Los servidores controlan el acceso a los recursos compartidos tales como sistemas de archivos, base de datos, acceso a redes WAN, impresoras, etc. Cada transacción del usuario tiene tres componentes en su tiempo de respuesta: la demora en la estación cliente, la demora en la red y la demora en el servidor.

En la estación cliente, se tiene en cuenta el tiempo de CPU y de disco para generar la solicitud. En la red se considera el tiempo necesario para el acceso a través del protocolo correspondiente y transmitir los paquetes que conforman la demanda del cliente a través de la red. La demora en el servidor está compuesta por el tiempo de servicio en CPU y de los discos y el tiempo de espera en cola para acceder a estos recursos. Las demoras en el tiempo de respuesta dependen de los recursos involucrados, sus características y de las cargas de trabajo o tipos de demandas de los usuarios en las estaciones cliente.

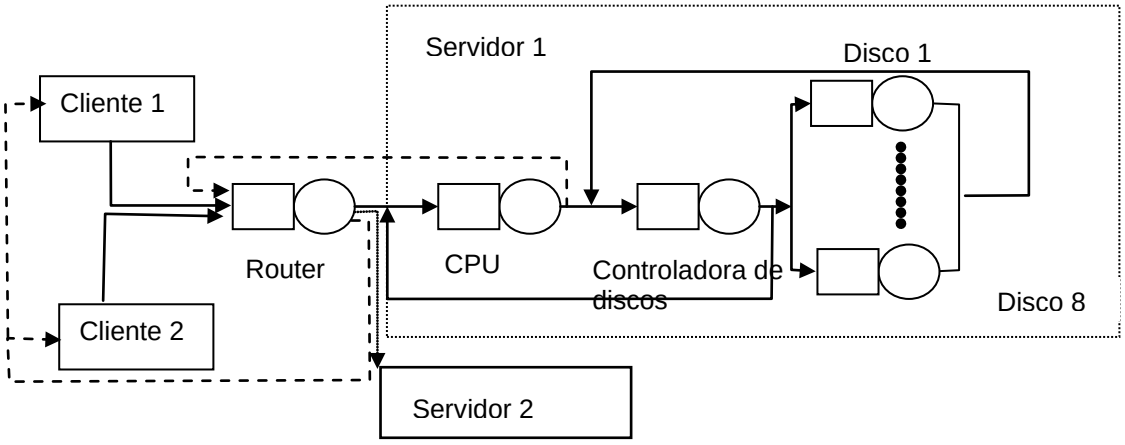


Figura 2: Modelo Cliente/Servidor con descripción detallada de estructura del servidor

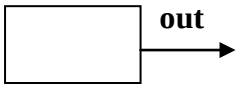
En la Figura 2, el camino de las peticiones generadas por los clientes a alguno de los dos servidores se representa a través de líneas discontinuas. El mensaje pasa a un router, encargado de direccionar los mensajes, ya sean éstos respuestas o solicitudes, hacia el destino correspondiente. Debido a que el router no responde de manera inmediata, el mismo posee una cola asociada, en donde se almacenan las peticiones de los clientes y las respuestas de los servidores, esperando a ser enviadas a su destino.

Los mensajes destinados a cualquiera de los servidores llegan a la CPU respectiva, en donde se encolan hasta ser atendidos. Desde la CPU, la solicitud puede requerir información almacenada en los discos del servidor. El sistema aquí propuesto tiene un arreglo de hasta 8 discos con una controladora encargada de enviar la petición según el disco seleccionado o el disco desocupado. Dicha controladora tiene también una cola asociada debido a que la misma sólo puede atender una petición a la vez.

Los discos también poseen una cola asociada, ya que la controladora puede enviar varias solicitudes al mismo disco antes que éste se desocupe, evitando el riesgo de pérdida. Una vez que una solicitud a disco es atendida, regresa a la CPU a través de la controladora. En la cola de trabajos de la CPU se le da prioridad a los requerimientos provenientes de los discos para completar los trabajos en ejecución, a fin de optimizar el tiempo de respuesta del sistema. Una vez completado el procesamiento de la petición, se envía la respuesta al cliente que la generó, a través del subsistema de comunicaciones (router, red).

A continuación, se describen las generalidades de los componentes DEVS atómicos que conforman el sistema modelado. A modo ilustrativo, se incluye la especificación en seudocódigo del modelo atómico "Cliente".

Modelo Atómico Cliente



El modelo Cliente es esencialmente un generador de solicitudes hacia los servidores, de acuerdo a una distribución determinada (especificada como parámetro del modelo). Este modelo fue programado para enviar solicitudes a dos servidores, a los cuales se le asignan las direcciones 0 y 1. La proporción de solicitudes hacia un servidor con respecto al otro es también parámetro modelo. Además, cada Cliente posee una dirección específica, la cual es utilizada luego por el router para encaminar las respuestas a la estación correspondiente.

A continuación, se detallan los parámetros de este modelo, así como también la especificación de las funciones que definen el comportamiento del cliente.

Parámetros:

distribution : distribución de probabilidad del tiempo entre solicitudes
initial: Valor inicial del ID de tarea
increment: Incremento del ID de tarea
address: Dirección del Cliente (2 – 15)
cargaserver1: Proporción (%) de carga del servidor 1 respecto del servidor 2

Función de Transición Externa

```
MostrarMensajeError();// Los clientes no deberían recibir entradas
PararSimulación()        // entonces mostrar mensaje de error
```

Función de Transición Interna

```
Esperar(Tiempo_al_Azar);    //Mantener el modelo en espera hasta que llegue el momento de
                                                    enviar  la próxima solicitud
```

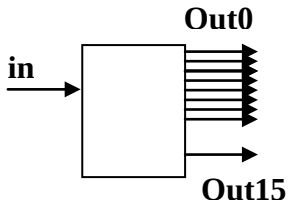
Función de Salida

```
Enviar(Petición_Actual, out); // Enviar petición por el puerto out
Petición_Actual = Preparar_Petición();
```

Función de Inicialización

```
Petición_Actual = Preparar_Petición();
Esperar(Tiempo_al_azar);
```

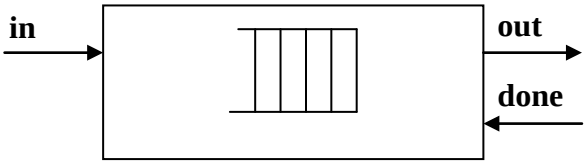
Modelo Atómico Router



En esta implementación del modelo Router, éste recibe paquetes de información (peticiones y/o respuestas) desde los distintos elementos de la red y los "encamina" a los distintos puertos de salida, de acuerdo a las direcciones de destino (out0 a out15), permitiendo conectar simultáneamente hasta 16 dispositivos (2 servidores y 14 clientes). Las direcciones de origen y destino son establecidas - como

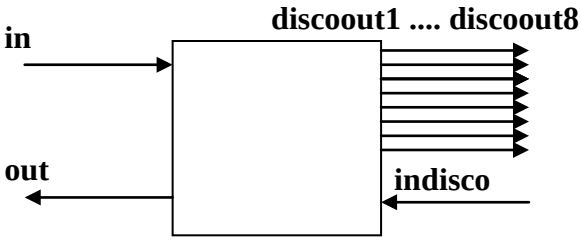
parámetro- en los clientes y servidores. El Router se asocia a una cola en donde se almacenan los paquetes hasta su envío. De esta manera, se evita la pérdida de paquetes en caso de que el Router se encuentre ocupado.

Modelo Atómico Cola



El modelo Cola consiste en una cola FIFO simple, la cual es alimentada a través del puerto in. Los paquetes o peticiones son enviados al exterior por el puerto out. Cuando la cola envía un elemento, espera la confirmación del receptor para enviar el siguiente. Dicha confirmación es recibida por el puerto done, luego de lo cual se programa el próximo envío, con una demora configurable por el modelador.

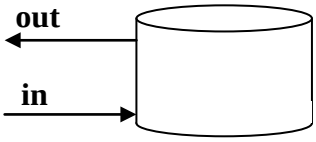
Modelo Controladora



El modelo controladora realiza el despacho de las solicitudes a los discos de los servidores, indicando a la CPU cuando las peticiones terminan su proceso. El envío de las peticiones a disco se realiza sin demora. Es decir, cuando arriba una solicitud o notificación de terminación, se envía inmediatamente a su destino (uno de los discos o la CPU). En la controladora también se decide el tamaño de la petición, el cual sigue una distribución de probabilidad, definida por el usuario.

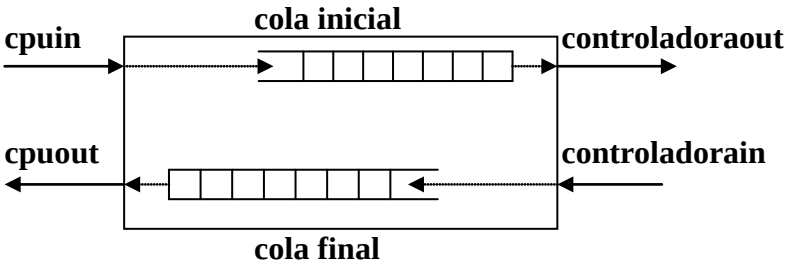
El disco de destino se elige al azar, siguiendo una distribución uniforme. Cabe destacar que el número de discos a acoplar a este modelo puede variar entre uno hasta un máximo de ocho. Esta cantidad es definida por el usuario.

Modelo Atómico Disco



El modelo disco recibe peticiones desde el exterior (en este caso, la cola de espera del disco), a través del puerto in y las atiende una a una. El disco permanece ocupado por un intervalo de tiempo que sigue una distribución de Poisson, cuya media depende del tamaño de cada petición. En caso de arribar una petición cuando el disco está ocupado con otra, se desaloja la anterior para atender a la nueva. Es por eso que debe asegurarse que las peticiones lleguen cuando el disco está libre. Cuando la petición se termina de procesar, ocurre una transición interna y se envía la petición terminada hacia el exterior, generalmente la controladora de disco.

Modelo Atómico: CPU



El modelo atómico CPU recibe solicitudes por el puerto cpuin**,** y realiza un procesamiento inicial. En un sistema real, este procesamiento podría corresponder, por ejemplo, a dar formato a los datos de entrada. Una vez completado dicho procesamiento, se envía una solicitud al subsistema de discos por el puerto controladoraout. En esta

implementación, la CPU realiza sólo una petición a disco. Futuras versiones del modelo permitirán iterar las llamadas a disco.

Como las solicitudes pueden arribar a intervalos aleatorios, las nuevas solicitudes se encolan en una "cola inicial", en donde aguardan a ser atendidas. Cuando las solicitudes provenientes del subsistema de discos arriban a la CPU, se encolan en una "cola final", en donde esperan a recibir un procesamiento final (formatear los datos de salida, por ejemplo), antes de ser enviadas fuera de la CPU por el puerto cpuout.

Cuando la CPU termina de procesar una petición, toma la próxima petición de la "cola final". Si no hay peticiones esperando procesamiento final, entonces toma una de la "cola inicial". Si ambas colas están vacías, el modelo entra en estado pasivo, hasta que arribe el siguiente trabajo. Los tiempos medios de procesamiento inicial y final son parametrizables.

Descripción de la herramienta gráfica (Visual DEVS)

La herramienta gráfica para la simulación de modelos de eventos discretos descriptos mediante el formalismo DEVS utiliza el motor de simulación CD++ [1] desarrollado por el Dr. Gabriel Wainer, del Departamento de Computación de la Universidad de Buenos Aires (UBA).

La interfaz multidocumento (MDI) incluida en esta herramienta permite trabajar en forma independiente con varios modelos a la vez. Las especificaciones de los modelos atómicos (parámetros, puertos de entrada/salida) se guardan en una base de datos, de la cual se extraen dichas especificaciones a la hora de agregar componentes a un modelo.

La finalidad del entorno es facilitar el diseño de modelos DEVS acoplados, de manera gráfica, sin necesidad de utilizar descripciones formales basadas en archivos de texto. El sistema permite agregar modelos y vínculos (entre modelos) de manera dinámica, a la vez que permite agregar nuevas descripciones de componentes atómicos.

La herramienta permite simular los modelos creados, y especificar los parámetros de la simulación: duración de la corrida, precisión de los valores numéricos, archivos de salida, etc. En la figura 3, se muestra una pantalla típica de la herramienta.

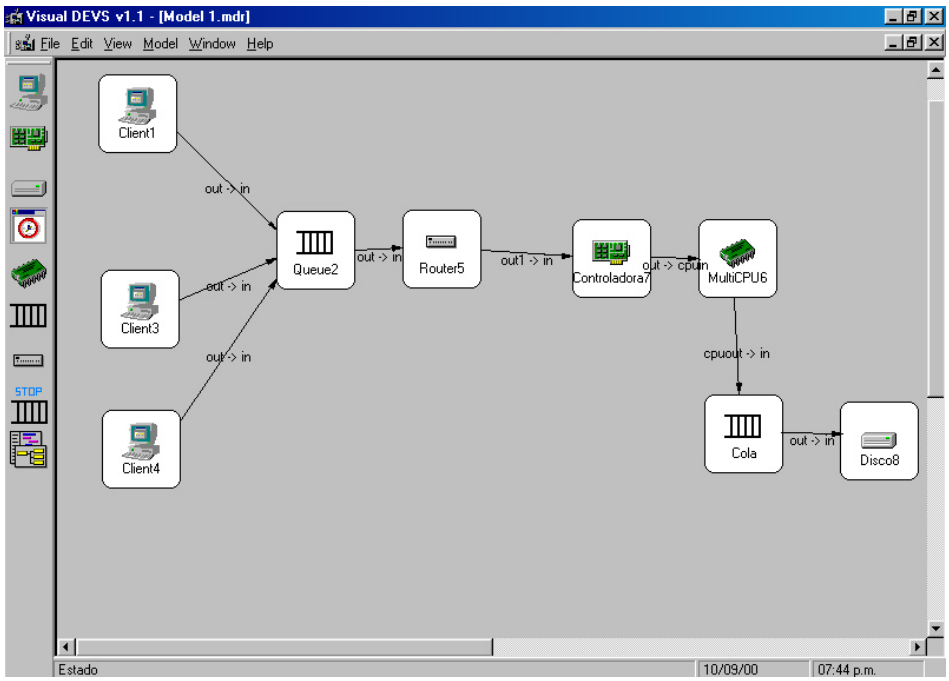


Figura 3: Entorno gráfico de la herramienta Visual DEVS

Resultados

Las variables de respuesta del modelo de simulación elegidas son la velocidad de procesamiento del sistema, el tiempo de respuesta global y la cantidad promedio de trabajos o longitud de cola en el sistema.

En el Caso 1 se hace una comparación entre el uso de uno y dos servidores. Las variables de entrada del sistema son el tiempo de procesamiento de la/s CPU/s, manteniendo constantes el resto de los parámetros del sistema.

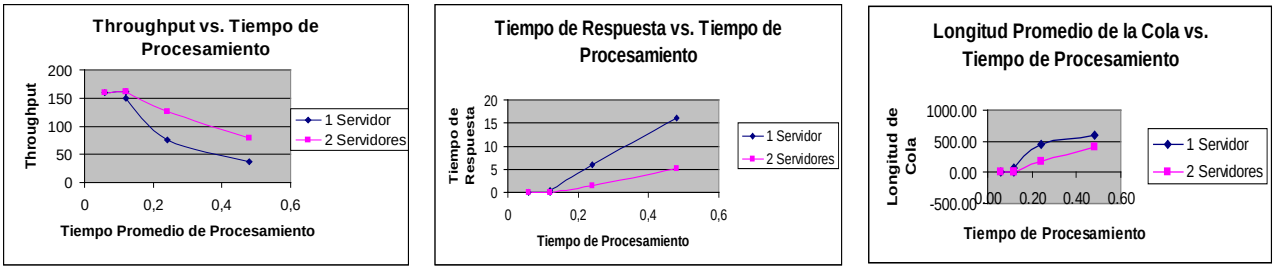


Figura 4: Indices de performance del sistema en función del tiempo de procesamiento.

Tal como podría esperarse, los gráficos demuestran que en ambos casos (1 y 2 servidores), la velocidad de procesamiento de los trabajos disminuye a medida que el tamaño de los trabajos (tiempo requerido para procesarlos) crece, a la vez que el tiempo de respuesta del sistema tiende a crecer en forma lineal. El tamaño promedio de cola, a medida que aumenta el tiempo de procesamiento, tiende a ser constante, lo cual sugiere que el nivel de peticiones actual no llega a saturar el sistema.

En el Caso 2 se comparan los índices de performance del Caso 1, esta vez variando el tiempo promedio entre peticiones, permaneciendo los demás parámetros iguales que el caso anterior. Los gráficos a continuación muestran los resultados obtenidos.

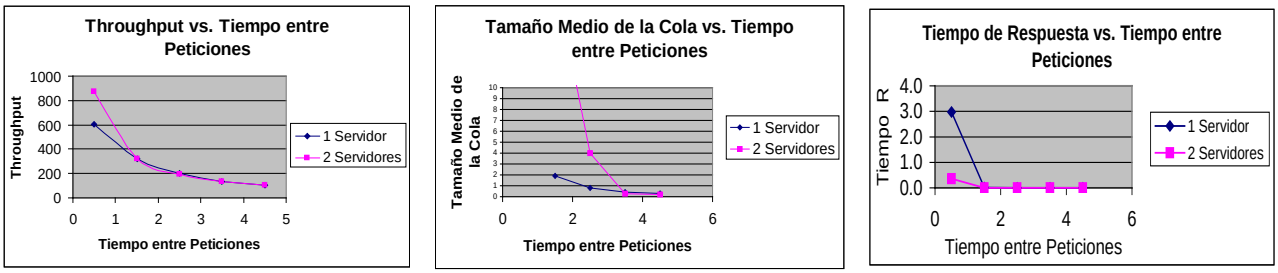


Figura 5: Indices de performance del sistema en función del tiempo entre peticiones.

Los gráficos de arriba muestran que no hay diferencias significativas en el la velocidad de procesamiento para pocas peticiones, tanto para el caso de un servidor como el que se utilizan dos. Si embargo, a mayor frecuencia de peticiones, la diferencia entre el uso de uno y dos servidores se hace más evidente. El tiempo de respuesta tiene un comportamiento similar al del throughput (a mayor nivel de peticiones, mayor la diferencia entre los casos de 1 y 2 servidores). En cuanto a la longitud media de cola del sistema, se observa una tendencia de crecimiento no lineal y una marcada diferencia entre el uso de uno y dos servidores.

En el caso 3 se considera la variación en el número de discos de los servidores, manteniendo constantes los demás parámetros del sistema.

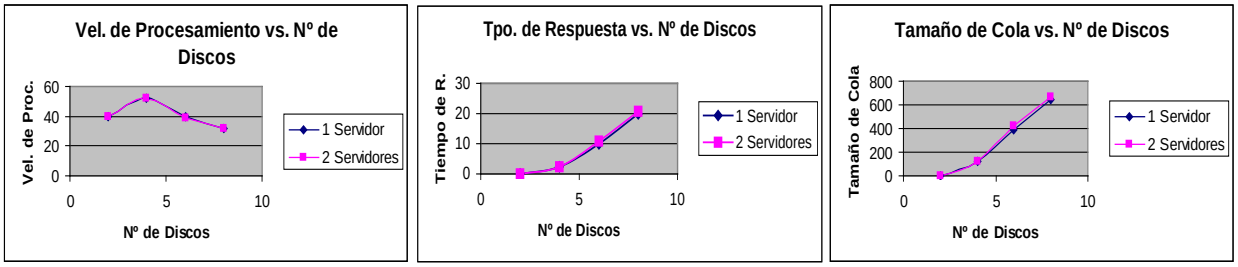


Figura 6: Indices de performance del sistema en función del número de discos de los servidores

De la observación de los resultados en los gráficos de las figuras 4, 5 y 6 se puede deducir que el cuello de botella del modelo está conformado por la CPU y el subsistema de comunicaciones, siendo las respuestas insensibles a las variaciones del subsistema de discos del/los servidor/es.

En el caso 4 se estudia el impacto en la performance del sistema provocado por la variación del retardo en el Router, manteniendo los demás parámetros constantes. En este caso, se utilizó sólo un servidor, ya que no había diferencias significativas, al usar uno o dos servidores, cuando se variaba el tiempo entre peticiones.

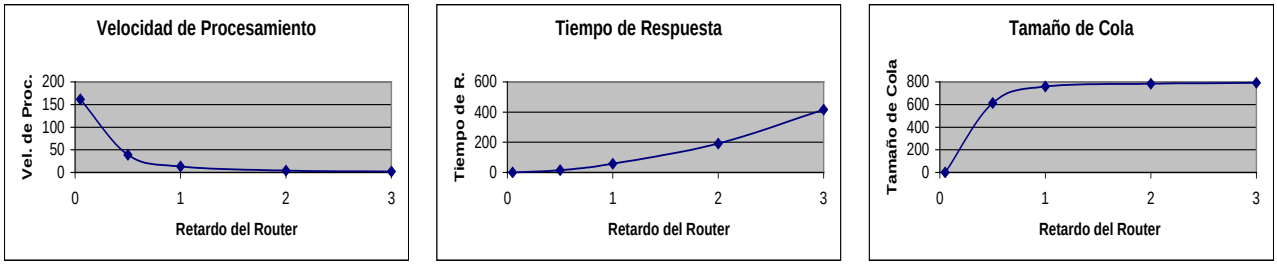


Figura 7: Indices de performance en función del tiempo de procesamiento del router

Del análisis de los resultados se deduce que, para un tiempo entre peticiones constante igual a 4.5 segundos, el tiempo de servicio del router tiene un notable impacto en la performance del sistema, porque a medida que éste aumenta, todos los indicadores de performance se degradan marcadamente.

Conclusiones

Se han desarrollado modelos de simulación de eventos discretos para redes de computadoras, especificando cada componente mediante el formalismo DEVS.

- ✓ De la simulación de estos modelos se obtuvieron resultados consistentes con las medidas de performance de sistemas cliente/servidor típicos.
- ✓ La utilización del formalismo DEVS ha probado ser un medio eficaz para el desarrollo modular de modelos de simulación. Además, la utilización de esta metodología ha permitido en forma sencilla la rápida modificación de los modelos, para adaptarlos a los distintos escenarios propuestos.
- ✓ El uso de herramientas gráficas (como Visual DEVS) es un complemento eficaz para el desarrollo de modelos DEVS, ya que potencia las bondades del formalismo evitando la necesidad de especificar las relaciones entre componentes mediante el volcado de dichas relaciones a una representación textual. Además, al presentar el modelo de manera gráfica, facilita la visualización global del sistema a modelar, mejorando la comprensión del mismo.

Trabajo Futuro

Este trabajo describe las tareas realizadas por el grupo de investigación GIRED, en el área de modelado y simulación de redes mediante el formalismo DEVS. A continuación se detallan las líneas de investigación en el tema:

- ✓ Se prevé una serie de mejoras a esta primera versión de la herramienta de interfaz gráfica, tales como el uso de modelos acoplados para realizar el modelado jerárquico del sistema, la inclusión de modelos celulares a la herramienta, y otra serie de facilidades.
- ✓ Actualmente se está definiendo los lineamientos generales de un trabajo en conjunto con otras Universidades Nacionales, con el fin de integrar las herramientas de modelado y simulación generadas en cada uno de los grupos de investigación.
- ✓ El próximo paso es realizar mediciones de performance en sistemas reales para obtener datos de entrada que alimenten los modelos generados, y así perfeccionar y validar los mismos.
- ✓ Luego de completadas las etapas anteriores, se prevé la extensión del área de estudio para incluir otros tipos de redes de computadoras, nuevos modelos de carga de trabajo y predecir el impacto en los índices de performance producidos por cambios en las configuraciones y/o requerimientos a los sistemas mencionados.

Bibliografía

[1] N-CD++ Manual del Usuario, Daniel A. Rodríguez y Gabriel A. Wainer, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina, 1999.

[2] Specification, modelling and simulation of timed Cell-DEVS spaces, Gabriel A. Wainer, Norbert Giambiasi. Report n.: 97-006

[3] Application of the Cell-DEVS paradigm. Gabriel A. Wainer; Amir Barylko; Jorge Beyoglonián; Norbert Giambiasi. Report n.: 99-018.

[4] WAINER, Gabriel, Departamento de Computación Fac. Ciencias Exactas y Naturales, Universidad de Buenos Aires, 1996

[5] ZEIGLER, B. "Theory of modeling and simulation". Wiley, 1976.

[6] ZEIGLER, B. "Multifaceted Modelling and discrete event simulation". Academic Press, 1984.

[7] ZEIGLER, B. "Object-oriented simulation with hierarchical modular models". Academic Press, 1990.

[8] ZEIGLER, B.; KIM, D. "Design of high level modelling / high performance simulation environments". Technical Report, Department of Electrical and Computer Engineering, University of Arizona. 1995.