Comunicación síncrona de simulaciones interactivas desarrolladas con Easy Java Simulations

Carlos A. Jara, Francisco A. Candelas, Fernando Torres
Dept. de Física, Ingeniería de Sistemas y Teoría de la Señal
Universidad de Alicante
03080, San Vicente del Raspeig
cajb@dfists.ua.es, Francisco.Candelas@ua.es, Fernando.Torres@ua.es

Francisco Esquembre
Dept. de Matemáticas
Universidad de Murcia
30071, Murcia
fem@um.es

Sebastián Dormido
Dept. de Informática y Automática
Universidad Nacional a Distancia
28040, Madrid
sdormido@dia.uned.es

Resumen

Con el principal objetivo de unir las ventajas de los laboratorios virtuales y los entornos colaborativos *on-line*, se presenta en este artículo un sistema de comunicación *síncrono* entre simulaciones interactivas desarrolladas en *Easy Java Simulations* (Ejs) a través del protocolo TCP/IP. De esta manera, un grupo de alumnos de una clase virtual coordinados por un profesor, puede experimentar con la misma simulación de forma on-line compartiendo experiencias y experimentos. En el contenido del artículo se describirán las principales características del sistema de comunicación y de su arquitectura, así como las posibilidades de sincronización de las simulaciones interactivas.

1. Introducción

La irrupción de las Tecnologías de la Información y la Comunicación (TIC) en nuestra sociedad ha sido espectacular, y desde el mundo educativo, no podemos ser ajenos a ello. Evidencia de este acontecimiento es la aparición de nuevas formas de enseñanza y aprendizaje, tales como los *laboratorios virtuales*. Este innovador objeto de aprendizaje es usado frecuentemente en la enseñanza virtual universitaria, sobre todo en las carreras y cursos de contenido técnico [2]. Los laboratorios virtuales cubren desde simples simulaciones de fenómenos físicos [11], hasta aplicaciones que permiten el acceso remoto a los recursos de un sistema real [3], [6].

Entre las múltiples ventajas que ofrecen los laboratorios virtuales dentro del campo educativo, cabe destacar que:

- Son herramientas de auto-aprendizaje, donde el alumno puede configurar nuevos experimentos permitiendo obtener una visión más intuitiva de los fenómenos.
- Acercan y facilitan a un mayor número de alumnos la realización de experiencias sin riesgo a dañar los equipos, flexibilizando sus horarios y evitando el solapamiento con otras asignaturas.
- Reducen el coste del montaje y mantenimiento de los laboratorios tradicionales, siendo una alternativa barata y eficiente, donde el estudiante simula los fenómenos a estudiar como si los observase en la realidad.

Sin embargo, la mayoría de los laboratorios virtuales son diseñados para ser usados de forma individual (*laboratorio virtual monolítico*), y no permiten el trabajo en grupo ni la colaboración entre los estudiantes y el profesor.

Los autores de este artículo, motivados por determinar el impacto de los laboratorios virtuales en la docencia universitaria, realizaron una evaluación de los mismos en asignaturas de contenido en Robótica, Visión Artificial y Automatización. La principal conclusión de este estudio fue que el estudiante valora muy positivamente la docencia virtual, y que los laboratorios virtuales le ayudan a comprender mejor los conceptos. Sin embargo, consideran los laboratorios virtuales como un apoyo a la docencia, no como una alternativa, ya que preferían asistir al laboratorio real, compartir sus problemas con los demás estudiantes y tener el apoyo de un profesor [4].

Durante los últimos años, se han desarrollado muchos entornos web de aprendizaje donde se ha incluido la colaboración y la comunicación.

Actualmente, podemos distinguir dos tipos de entornos colaborativos: los *on-line* o *síncronos* y los *off-line* o *asíncronos*. Con respecto a los síncronos, son los entornos que más se aproximan a la enseñanza tradicional, ya que permiten a los estudiantes compartir experiencias en tiempo real durante la clase virtual. En cuanto a los asíncronos, los integrantes de la clase no tienen porque estar conectados al mismo tiempo al sistema, aportando gran flexibilidad en el horario.

La mayoría de los entornos colaborativos on-line existentes en la actualidad, utilizan herramientas clásicas para la comunicación entre los integrantes de la clase tales como *chats*, *video streams*, pizarras compartidas [10] o incluso con paneles de dibujo interactivos [12]. Sin embargo, dichas herramientas limitan el aprendizaje a la hora de explicar conceptos de carácter técnico, que podrían ser observados más fácilmente a través de las simulaciones de un laboratorio virtual. Aun así, el principal problema que existe actualmente es que hay muy pocos entornos colaborativos de carácter síncrono.

Entre uno de los entornos asíncronos más importantes, está *eMersion* [9]. Este sistema permite el acceso individual a laboratorios virtuales y remotos a través de la web. Además, dentro de este entorno se ha integrado *eJournal*, espacio electrónico donde los usuarios pueden compartir documentos y resultados de experimentos.

A día de hoy, existen entornos síncronos donde se ha intentado combinar laboratorios virtuales y herramientas clásicas de comunicación [1]. Pero han resultado ser entornos complejos y la necesidad de utilizar software costoso para implementar los laboratorios virtuales, como por ejemplo *Matlab Web Server*.

Considerando todo lo expuesto en las líneas anteriores, nuestro grupo de investigación ha creado un sistema de comunicación síncrono a través del protocolo TCP/IP entre simulaciones Java desarrolladas con Ejs, para ser integradas dentro de entornos colaborativos on-line y realizar la enseñanza en tiempo real a través de Internet.

El artículo se ha organizado de la forma siguiente. Primero, se explican brevemente las características generales de las simulaciones creadas con Ejs. Posteriormente, se describe los aspectos más importantes del sistema de comunicación. En la sección 4 se expone cómo son sincronizadas las simulaciones. Finalmente, se comentan algunas conclusiones y posibles trabajos futuros.

2. Simulaciones interactivas *Ejs*

Ejs es una herramienta de software *open-source* diseñada para la creación de simulaciones dinámicas interactivas en lenguaje Java [8]. Ejs es parte del proyecto *Open Source Physics*, que se estableció con el objetivo de crear y distribuir simulaciones para su uso pedagógico en la enseñanza de la física [5]. Las simulaciones creadas con Ejs pueden ser usadas como programas *stand-alone* bajo diferentes sistemas operativos o ser distribuidas como *applets* a través de Internet.

Las simulaciones Ejs constan de dos partes bien diferenciadas: el modelo y la vista. El modelo está constituido por las variables de la simulación, su valor inicial y por las ecuaciones matemáticas que rigen la evolución del sistema. La vista es la interfaz de usuario donde se muestra la representación gráfica de los diferentes estados del sistema. Ambas partes se encuentran interconectadas. Cualquier cambio en el estado del modelo es visualizado en la vista, y si el usuario interacciona con ella (controles de la interfaz), es capaz de modificar el valor de una variable del modelo.

Durante la ejecución de la simulación, Ejs crea las variables, las inicializa y ejecuta las ecuaciones del modelo cada cierto diferencial de tiempo para alcanzar un nuevo estado. Esta última fase es conocida como *paso de simulación* o *step*. Posteriormente, Ejs actualiza la vista para visualizar el nuevo estado alcanzado y vuelve a ejecutar las ecuaciones para continuar la evolución del sistema.

Unas de las propiedades más interesantes que posee Ejs es la interactividad de sus simulaciones [7]. Cada vez que el usuario pulsa un control o interacciona con algún objeto de la vista, se producen eventos que la simulación se ocupa de captar y gestionar. El usuario es el encargado de darle funcionalidad a estos eventos. Para esta tarea, Ejs proporciona una serie de métodos predefinidos entre los que hay que destacar los siguientes:

- *_play()*. Ejecuta la simulación. Es decir, da *steps* cada diferencial de tiempo.

- *_pause()*. Detiene la evolución de la simulación.
- *_reset()*. Detiene la simulación y la lleva a su estado inicial.
- *_update()*. Actualiza las variables del modelo.

Existen muchos más métodos predefinidos, pero los nombrados son los más importantes para la gestión de la simulación.

3. Comunicación entre las simulaciones

Esta sección describe cuáles son los *roles* de cada uno de los componentes de la clase virtual: profesor y alumno. También se comenta su funcionalidad y las posibilidades de cada uno. Finalmente, se explica el porqué del protocolo de comunicación utilizado y la arquitectura del sistema.

3.1. Componentes y funcionalidad del sistema

Como se ha comentado anteriormente, la clase virtual se compone de la simulación del profesor (*maestro*) y de las simulaciones de los alumnos (*esclavos*).

El maestro es el encargado de gestionar la sesión y la evolución de la simulación. Con respecto a la sesión, el maestro posee una lista de los alumnos conectados (Figura 1), a los que en cualquier momento puede desconectar. Referente a la simulación, es el único que puede interactuar sobre ella en un primer momento y comunicar a los alumnos conectados el estado de la misma.

Figura 1. Lista de alumnos

Para el comienzo de la sesión, el maestro debe activar el modo colaborativo (*estado de escucha*). A partir de este instante, los alumnos que deseen podrán conectarse al maestro, con la consecuente deshabilitación de todos los controles de su simulación para no poder interactuar con ella (Figura 2). Todos los alumnos conectados en la misma sesión, visualizarán en su simulación lo que el profesor realice sobre la suya.

Otra funcionalidad importante es la asignación de la *tiza* (ver punto 4.3). Con ella el maestro da a la simulación de un alumno el permiso de llevar el control sobre todas las demás. Tan sólo son permisos para la simulación, ya que la gestión de la sesión siempre queda en manos del maestro.

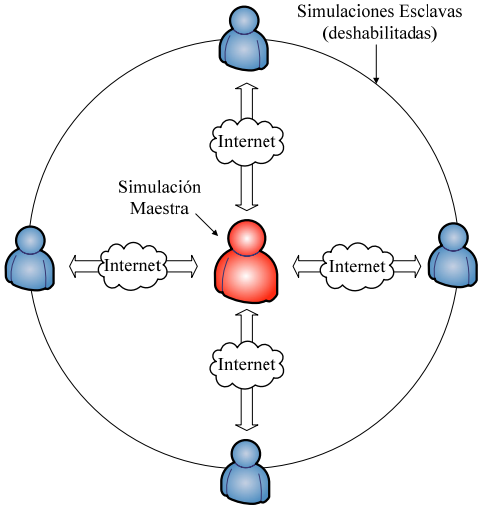

Figura 2. Componentes del sistema

3.2. Protocolo de comunicación

El sistema de comunicación entre las simulaciones está basado en *TCP sockets* (Figura 3). Durante el diseño del sistema, se planteó la posibilidad de utilizar protocolos de más alto nivel, como por ejemplo HTTP. Esta opción permitía evitar posibles problemas con los *firewalls*. Sin embargo, se desechó fundamentalmente por dos razones:

1. Por la necesidad de tener instalado en el maestro un software para dar servicios como servidor de Internet (*ISS*, *Apache*, etc,..).

2. Por el procedimiento de petición-respuesta del protocolo en sí, que complicaba la sincronización entre las simulaciones.

Mediante el protocolo TCP/IP estos problemas son evitados, y si además realizamos la comunicación por el puerto 80, también conseguimos evitar los posibles problemas con los firewalls, que era la principal ventaja de HTTP.

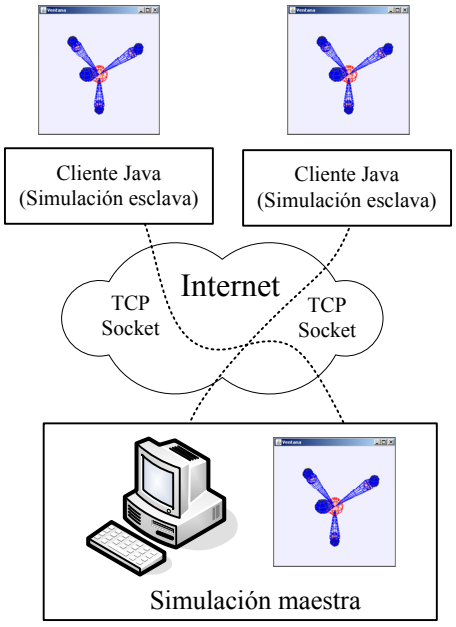

Figura 3. Sistema de comunicación

El principal problema en el uso de este protocolo para la comunicación entre las simulaciones, es en el caso en que los alumnos se encuentren en redes con conexiones WAN deficientes. Esta dificultad aumenta el retardo en la conexión y consecuentemente ralentiza la ejecución de la simulación. En cualquier caso, siempre se mantienen sincronizadas.

3.3. Arquitectura del sistema

Dado que la funcionalidad de la simulación del profesor y la del alumno son distintas, es necesario diferenciar sus arquitecturas.

Para iniciar el sistema de comunicación, el maestro debe activar el modo colaborativo. A partir de este momento un *SocketServer Java* abre un puerto determinado y comienza la escucha de peticiones de los alumnos (Figura 4). Este proceso de escucha es el responsable de la conexión y desconexión de las simulaciones de los alumnos.

Por cada petición aceptada, se crea un elemento más en un vector de conexiones (*sockets* esclavos), además de actualizar la lista de alumnos. Dicho elemento, contiene información referente a la procedencia del alumno (IP y puerto) conectado a la clase virtual. El objeto *repetidor* es el encargado de enviar los datos del estado actual de la simulación a todos los elementos de dicho vector. Cada cambio o evento en la simulación maestra es comunicado al repetidor, que se encarga de ponerlo en la red y hacerlo llegar a los clientes IP.

Además, para poder conocer el estado de cada una de las simulaciones de los alumnos, se establece un *lector* por cada conexión aceptada. Dicho objeto se encarga de comunicar al maestro si el esclavo ha realizado correctamente el paso de simulación que le envió el repetidor. Una vez que se han recibido todas las confirmaciones de los esclavos, el maestro puede continuar con la evolución del sistema. Así, se consigue la sincronización deseada en las simulaciones.

En el caso de existir un esclavo con la tiza asignada, el maestro actúa como si fuera un alumno, además de comunicarle al resto las órdenes que recibe de dicha simulación.

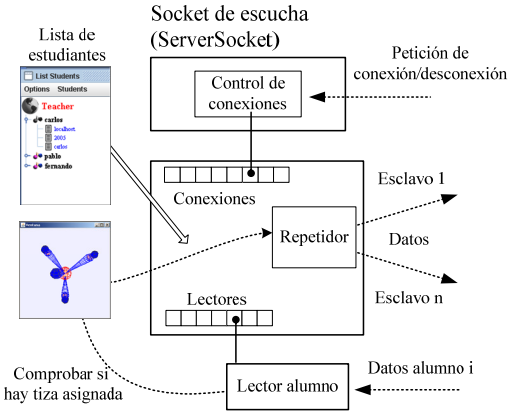

Figura 4. Arquitectura del maestro

La arquitectura del esclavo es mucho más sencilla que la del maestro (Figura 5). Una vez establecido el canal de comunicación (socket), sólo existe un intercambio de datos profesor-

alumno. Un proceso *receptor* se ocupa de recibir las órdenes del maestro para controlar y actualizar la simulación. Y un *remitente* de datos, comunica al lector correspondiente del maestro (ver Figura 4), el estado de la simulación.

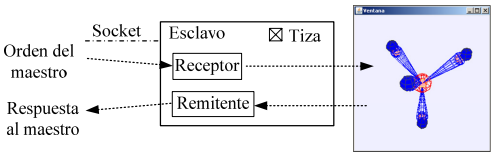

Figura 5. Arquitectura del esclavo

Si el esclavo tiene la tiza asignada, sus controles se habilitan para que el alumno pueda interactuar sobre ella y comportarse como si fuera el maestro de la clase virtual.

4. Sincronización *on-line*

El sistema de comunicación y sincronización de las simulaciones se ha incluido como una opción más del software Ejs en una versión experimental, de forma que se aplica a las simulaciones compiladas con esta versión. Desde una simulación Ejs que se encuentre en un ordenador con conexión a Internet, es posible ponerla a la escucha de peticiones de alumnos o conectarse a una simulación maestra para recibir una clase virtual.

Dado que se está trabajando con simulaciones idénticas, el modelo del sistema es el mismo para todos los componentes de la sesión, facilitando la tarea de actualización de variables y refresco de la vista.

Por el socket de comunicación establecido entre el profesor y el alumno, se envía objetos Java que normalmente contienen *Strings*. Aquí, es necesario distinguir entre dos tipos de mensajes:

1. Los correspondientes al paso de simulación.
2. Los correspondientes a eventos como: *_pause()*, *_reset()*, *_update()*, etc,…

4.1. Paso de simulación

Tal y como se comentó en la sección 2, el paso de simulación o step es la resolución del modelo del sistema en un diferencial de tiempo. Por tanto, para sincronizar las simulaciones, es necesario que todas se encuentren en el mismo step. Para ello, cuando el maestro comienza la ejecución de la simulación y da el primer paso de simulación, envía a través del repetidor la orden *"step"* para el resto de simulaciones. En este momento, para la ejecución y se pone en espera hasta que todas las simulaciones de los alumnos hayan realizado el step. De esta manera se consigue la sincronización, propiedad muy apta para el aprendizaje colaborativo on-line.

4.2. Eventos

En el caso de los eventos, el sistema de comunicación es más sencillo. Gracias a que Ejs capta y gestiona los eventos, tan sólo tenemos que decirle que envíe el mensaje cuando ese evento tenga lugar. Aquí, no es necesario esperar a que los clientes verifiquen que han ejecutado la orden.

4.3. Asignación de la tiza

Como se ha comentando en este artículo, el sistema de comunicación puede dar el permiso a un alumno de controlar el estado de la simulación.

El maestro sólo puede asignar la tiza a uno de los alumnos conectados. En el momento de la asignación, se desconectan los controles de la simulación maestra y se activan los de la simulación esclava que posee la tiza para poder controlarla.

Para la sincronización, no es necesario que la simulación con la tiza tenga la lista de alumnos, ya que todas las acciones que realice son comunicadas al maestro, que se encarga de retransmitirlas al resto de clientes conectados. En el caso del step, el esclavo que posee la tiza se espera a que el maestro le indique que todos los alumnos han dado el paso de simulación.

4.4. Resultados

En la Figura 6 podemos observar la sincronización entre tres simulaciones de un tiro parabólico ejecutadas en un mismo ordenador. La simulación más a la izquierda es la maestra (controles activos), mientras que las otras dos son las esclavas (controles desactivados). El campo numérico que se encuentra marcado es el tiempo de simulación, que como podemos observar es el mismo en todas.

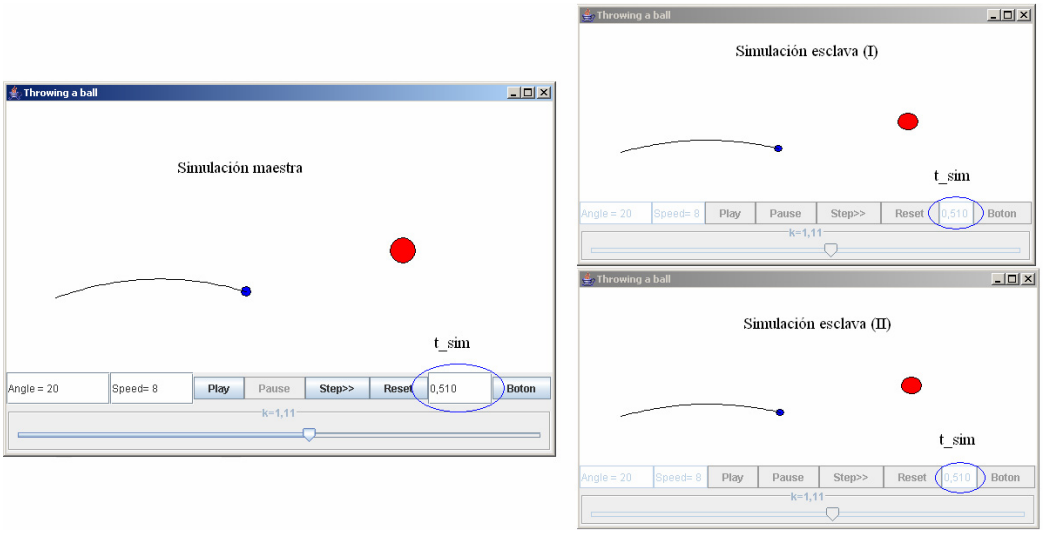

Figura 6. Simulaciones sincronizadas

El sistema de comunicación también se probó en una red local, con un número de tres PC con simulaciones esclavas conectadas a un PC maestro. La sincronización resultó ser más rápida que realizándola sobre un mismo ordenador.

5. Conclusión y trabajos futuros

La combinación de los laboratorios virtuales y los entornos colaborativos on-line es hoy en día la unas de las mejores alternativas para la educación a distancia. Los laboratorios virtuales son herramientas de auto-aprendizaje que ayudan a comprender mejor los conceptos a los alumnos. Y los entornos colaborativos on-line permiten compartir ideas y experiencias entre el profesor y los alumnos en tiempo real.

Muchos entornos colaborativos existentes en la actualidad incluyen laboratorios virtuales y remotos. Sin embargo, muy pocos de ellos son de carácter síncrono. Motivados por esta carencia, los autores han presentado en este artículo un sistema de comunicación on-line, usando TCP Sockets, entre simulaciones interactivas desarrolladas con Ejs, herramienta que ayuda a crear simulaciones dinámicas interactivas en el lenguaje Java.

El sistema se ha incluido como una opción más del software Ejs, con la ventaja de que cualquier simulación creada con esta herramienta y usada en un ordenador conectado a Internet, sirve para realizar una clase virtual. La sincronización de las simulaciones permite al profesor enseñar en tiempo real a todos los alumnos conectados.

Uno de los inconvenientes que presenta este trabajo es que el alumno necesita poseer el archivo de simulación y las librerías para su ejecución en su ordenador. A día de hoy, los autores trabajan para evitar este problema. Intentan que el sistema de comunicación genere una *URL* con un applet que contenga la simulación sincronizada. A esta dirección web se conectará el alumno sin necesidad de poseer en su ordenador la simulación.

Agradecimientos

El trabajo presentado en este artículo está financiado por el Ministerio de Educación y Ciencia a través del programa de becas FPI y el proyecto de investigación DPI2005-06222. Los autores quieren agradecer esta financiación.

Referencias

[1] Abler, R. T., Wells, I. G. Distributed Engineering Education: Evolution of the Telecollaboration Stations for Individualized Distance Learning. IEEE Transactions on Education, 2005.

[2] Candelas, F. A., Moreno, J. S. Recursos didácticos basados en Internet para el apoyo a la enseñanza de materias del área de Ingeniería de Sistemas y Automática. Revista Iberoamericana de Automática e Informática Industrial, 2005.

[3] Candelas, F. A., Puente, S. T., Torres, F., Ortiz, F. G., Gil, P., Pomares, J. A virtual laboratory for teaching robotics. International Journal of Engineering Education, 2003.

[4] Candelas, F. A., Torres, F., Ortiz, F., Gil, P., Pomares, J., Puente, S.T. Teaching and Learning Robotics with Internet Teleoperation. Second International Conference on Multimedia and Information & Communication Technologies in Education, 2003.

[5] Christian, W., Belloni, M. Developing open source programs for science and mathematics. Eurocon, 2003.

[6] Dormido, S., Esquembre, F. The Quadruple-Tank Process: An Interactive Tool For Control Education. Proceedings of the European Control Conference, 2003.

[7] Dormido, S., Farias, G., Sanchez, J., Esquembre, F. Adding interactivity to existing Simulink models using Easy Java Simulations. 44th IEEE Conference on Decision and Control, 2005.

[8] Esquembre, F. Easy Java Simulations: a software tool to create scientific simulations in Java. Computer Physics Communications, 2004.

[9] Gillet, D., Anh Vu Nguyen, N., Rekik, Y. Collaborative web-based experimentation in flexible engineering education. IEEE Transactions on Education, 2005.

[10] Kreutz, R. NetChat: Communication and Collaboration via WWW. Journal of Educational Technology & Society, 2000.

[11] Martín, C., Dormido, S., Pastor, R. , Sánchez, J. Sistema de Levitación Magnética: Un Laboratorio Virtual en Easy Java Simulation. XXIV Jornadas de Automática, 2003.

[12] Snow, C. Network EducationWare: An Open-Source Web-based System for Synchronous Distante Education. IEEE Transactions on Education, 2005.