Computers & Education 52 (2009) 126–140

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Real-time collaboration of virtual laboratories through the Internet

Carlos A. Jara [a,*], Francisco A. Candelas [a], Fernando Torres [a], Sebastian Dormido [b], Francisco Esquembre [c], Oscar Reinoso [d]

[a] Department of Physics, System Engineering and Signal Theory, University of Alicante, San Vicente del Raspeig, Spain
[b] Department of Computer Science and Automatic Control, UNED University, Madrid, Spain
[c] Department of Mathematics, University of Murcia, Murcia, Spain
[d] Department of Industrial Systems Engineering, Miguel Hernandez University, Elche, Spain

ARTICLE INFO

Article history:
Received 8 March 2008
Received in revised form 9 July 2008
Accepted 10 July 2008

Keywords:
Collaborative learning
Distance education and telelearning
Interactive learning environments
Simulations

ABSTRACT

Web-based learning environments are becoming increasingly popular in higher education. One of the most important web-learning resources is the virtual laboratory (VL), which gives students an easy way for training and learning through the Internet. Moreover, on-line collaborative communication represents a practical method to transmit the knowledge and experience from the teacher to students overcoming physical distance and isolation. Considering these facts, the authors of this document have developed a new dynamic collaborative e-learning system which combines the main advantages of virtual laboratories and collaborative learning practices. In this system, the virtual laboratories are based on Java applets which have embedded simulations developed in Easy Java Simulations (EJS), an open-source tool for teachers who do not need complex programming skills. The collaborative e-learning is based on a real-time synchronized communication among these Java applets. Therefore, this original approach provides a new tool which integrates virtual laboratories inside a synchronous collaborative e-learning framework. This paper describes the main features of this system and its successful application in a distance education environment among different universities from Spain.

1. Introduction

The first computer-supported collaboration system emerged in 1984 from the need for sharing interests among product developers and researchers in diverse fields (Grundin, 1994). This revolutionary approach was called computer-supported collaborative work and it was used to learn by means of desktop and video conferencing systems. Consequently, a new paradigm arose around educational institutions which was defined as computer-supported collaborative learning (CSCL). This emerging system was based on the contributions of constructivist learning theories about the term *collaborative learning*, which focus on social interdependence and maintain that students consolidate their learning by teaching one another (Alavi, 1994). CSCL environments were created for using technology as a mediation tool within collaborative learning methods of instruction (Koschmann, 1994). Since then, and thanks to the great evolution of network technologies, education is moving out of traditional classrooms.

Although the acronym CSCL was the original name (O'Malley, 1995), this term is also known as web-based collaborative learning because of the exponential spread of the World Wide Web over the last few years. These collaborative e-learning environments have caused a revolution in the academic community, providing a great amount of advantages for using both the Internet and technologies for 'any-time, any-place' collaborative learning. From a pedagogical and constructivist perspective, collaborative learning methods tend to encourage construction of knowledge, deeper understanding and greater skill development by their ability to engage students dynamically in the learning process (Alavi, 1994).

Nowadays, many academic institutions provide websites for e-learning as electronic repositories of knowledge and information, where teachers upload computer files, multimedia materials (audio and video files) and links to external knowledge sources. Students can access these materials at any time from anywhere and learn in his/her own way. In many cases, the management of these web environments (web-based publishing, format management, access rights and course registration) are provided by a content management system

* Corresponding author. Tel.: +34 96 590 94 91.
E-mail address: cajb@dfists.ua.es (C.A. Jara).

0360-1315/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2008.07.007

(CMS), computer software tool designed to make web administrators (in this case teachers) the creation and organization of their on-line courses easier. However, these systems do not support the management to plan, implement, and assess a specific learning process. Learning management systems (LMS) are software applications which meet this need providing tools to create and deliver content, monitor student participation and assess student performance. In addition, they also provide both collaboration tools such as e-mail, wikies, on-line forums, discussion boards, chats, etc. and the possibility to import collaborative activities based on pedagogical strategies such as Jigsaw or Pyramid (Hernandez et al., 1994). Therefore, on-line courses performed by LMS are usually e-learning platforms which include CSCL environments. Software tools such as webCT (Wheeler, 2000), Blackboard (www.blackboard.com) and Moodle (www.moodle.org) can be considered as LMS.

At the moment, there are two different types of CSCL environments according to the moment when the student–teacher interaction takes place: asynchronous and synchronous systems (Bafoustou & Mentzas, 2002). The first ones allow data exchange in flexible timetables and remote access to web-based course materials to carry out activities in an asynchronous way. They use collaborative tools such as e-mail or forums for on-line communication. However, this type of communication can cause feelings of isolation in the student and hence reduces his/her motivation (Kamel, Taylor, & Breton, 2005). In addition, students do not receive instant feedback from their questions and cannot talk in real-time about results obtained in the learning activities. These limitations have been solved by applying synchronous technologies (Marjanovic, 1999). CSCL synchronous environments enable the e-learning in a similar way to the traditional classrooms, sharing experiences and knowledge through the Internet in real-time like a face-to-face interaction. These systems usually use chats, audio and video conferences (Isenhour, Carroll, Neale, Rosson, & Dunlap, 2000; Kreutz, Kiesow, & Spitzer, 2000), shared desktop, shared whiteboard (Abler & Wells, 2005; Yang & Liu, 2007), and shared graphical applications (Vicent, Anguera, Golobardes, Badia, & Segarra, 2005). Among all the programs that provide these communication tools, it is worth mentioning Microsoft NetMeeting, Virtual Network Computing, Java Multimedia Tools and open-source software such as Globus Toolkit, Skype, Messenger, etc.

Nowadays, CSCL synchronous environments are mainly focused in the learning of theoretical lessons. However, sometimes theory does not provide enough knowledge to students, especially in the field of engineering education. As a solution to this problem, virtual laboratories (VLs) represent distributed environments of simulation which are intended to perform the interactive simulation of a mathematical model of a real system (Dormido, 2004). By means of them, students can learn through the Internet in a practical way and thus become aware of physical phenomena that are difficult to explain from just a theoretical point of view. In particular, interactive VLs are effective pedagogical resources, well suited for web-based and distance education. Their interactivity encourages students to play a more active role in the e-learning process and provides realistic hands-on experience (Dormido, Dormido, Sanchez, & Duro, 2005). Moreover, VLs are widely used for science teaching in areas such as Engineering, Physics, Mathematics or Biology since they provide everyone with public web sites to do practical experimentation from anywhere (Ma & Nickerson, 2006). Nevertheless, the majority of the VLs added in web-learning environments are designed to be used individually, and they do not allow work group or collaboration among students and teachers (Moreno, Gonzalez, Castilla, Gonzalez, & Sigut, 2007). The integration of VLs inside collaborative learning environments can be seen in *eMersion* (Gillet, Anh, & Rekik, 2005). This web-based platform contains a series of VLs whereby students can experiment and share results among other students or with teachers (Dormido et al., 2008). However, the collaboration in this CSCL environment is in an asynchronous way.

Bearing in mind the previous aspects, authors consider that several educational advantages could be achieved from combining VLs with a synchronous collaborative learning, especially for education of practical experiences in technical or science subjects. Like in a real laboratory environment or traditional classroom, the new synchronous collaborative e-learning system presented in this paper allows a group of students to share experiences at the same time they practice and explore experiments using VLs. This system also permits teachers to track, supervise and help students in their experimental exercises in a synchronous way. This can reduce strongly the teacher's tutorial process and allow students to resolve the experiments in a more guided way. In addition, this approach is supported by previous research done by the authors about the use of VLs in the educational process (Candelas et al., 2003; Candelas, Puente, Torres, Segarra, & Navarrete, 2005). The main results of this research stated that students consider classic asynchronous VLs as effective web-learning resources to the traditional teaching, but many of them prefer to perform their practical exercises in the real laboratory since they can work in coordination with their classmates and with the teacher support.

The proposed system combines the main advantages of VLs and the synchronous collaborative learning practice. The VLs are based on Java applets which have embedded simulations developed in *Easy Java Simulations* (EJS) (Esquembre, 2004), and the collaborative e-learning framework is a synchronized communication in real-time among the VLs. Thus, this novel approach integrates VLs inside a distance synchronous collaborative educational methodology. Teachers and students will be enabled to share practical experiences in an e-learning synchronous environment by means of VLs. In addition, the system can be applied to all the VLs developed in EJS in an easy way and can be used as a main tool in any CSCL environment. This supposes a portable collaborative framework for distance teaching and learning in several science disciplines, a feature which has not been implemented before in previous research projects.

The remainder of this paper is organized as follows: Section 2 discusses the related work which follows the synchronous CSCL approach. The following section describes basic aspects of EJS, which are necessary to understand the research presented. Next, the system architecture will be explained in detail. Afterwards, a complete example of a virtual collaborative class' generation will be shown in Section 5. A performance evaluation of the system is shown in Section 6. Some educational results are presented and discussed in Section 7. Finally, some important conclusions are shown in Section 8.

2. Background

One of the most meaningful problems of the distance collaborative tools is their platform dependence. Therefore, users who want to communicate through the Internet in heterogeneous environments have to install the suitable version for their operating system. NetMeeting by Microsoft can be one example of this, since it only runs under Windows and it is not able to communicate with any group of users. However, with the introduction of Java,[1] an advanced network programming language which is platform independent, has made it possible to

[1] Java Technology: http://java.sun.com.

overcome this problem. Java-based collaborative tools allow users the communication of distributed applications running in different platforms. In addition, Java applets are an alternative for stand-alone applications. They can be shared through a Java-enabled web browser without the user having to download and to install any specific software. In short, the use of Java has allowed the development of collaborative tools with a higher accessibility, portability and platform independence. The approach presented in this paper represents a Java-based collaborative tool which permits to share VLs embedded in Java applets in an easy and transparent way.

Nowadays, there are other platform independent environments such us Adobe Flash, Microsoft Silverlight and JavaScript. However, none of them are as powerful as Java to create both VLs and Internet distributed applications. Adobe Flash and Microsoft Silverlight are platforms usually used for adding interactivity to web pages (animations, videos, etc.) but they do not provide neither an extensive API nor suitable commands to develop the mathematical model of a simulation. Finally, JavaScript is a programming language used to make HTML documents more dynamics (pop up dialog boxes, open new windows, display messages, etc.) but it lacks of the extension and flexibility of the Java programming.

Since the appearance of Java language, many web-and-Applet-based collaboration tools have emerged to improve CSCL environments, but none of which are able to share VLs as explained in this paper. Moreover, most of them require the use of an Application Programming Interface (API), with the additional cost of modifying and implementing source-code to make Java applets collaborative. Among them, it is worth pointing out the following: Kuhmünch, Fuhrmann, and Schöppe (1998) developed a Java Remote Control Tool which allows the control and synchronization of distributed Java applets. The Habanero framework (Chabert, Grossman, Jackson, Pietrowiz, & Seguin, 1998) is a programming tool that supports the development of real-time collaboration environments. Java Collaborative Environment (JCE) is an extended version of the Java-AWT called Collaborative AWT, where mouse and keyboard events are intercepted and distributed among all the shared Java applets (Abdel & Kvande, 1997). Finally, Java Shared Data Toolkit (JSDT[2]) from Sun Microsystems is also an API-based framework used to add collaboration features to Java applets. With these approaches, besides having to modify the source-code of Java applets, users have to install first the Java Development Kit (JDK[3]), a software package for Java developers, and then a Java Integrate Development Environment (IDE) such as Eclipse (www.eclipse.org) or JCreator (www.jcreator.com), popular software tools to edit and program Java applications. In contrast, the system presented in this paper is not a new API and users do not have to modify any source-code of the Java files to create a collaborative environment with Java applets. In addition, users only need a web browser and the Java Runtime Environment (JRE) installed. To install the Java plug-in JRE, users can access to this Java Sun website,[4] download and install any version from JRE 1.4 to JRE 1.6.

There are other systems which do not propose a new API for developing collaborative environments. This is the case of JETS (Shirmohammadi, De Oliveira, & Georganas, 1998) and JASMINE (Shirmohammadi, El Saddik, Georganas, & Steinmetz, 2003). JETS (Java Enabled Telecollaboration System) is a client-server framework which allows sharing Java applets. However, it can only share some specific multimedia applications: a whiteboard, a VRML viewer and an on-line video. JASMINE (Java Application Sharing in Multi-User INteractive Environments) is a system which allows users sharing almost any Java applet on the Internet. This client-server framework captures both AWT-based and Swing-based events from the user interaction and sends them to all other participants connected to the server session. However, this approach cannot be applied to VLs, since these web e-learning resources often describe physical phenomena where some events and some updating of variables may happen without a user interaction.

It is also possible to find other applications which use collaborative applets for implementing VLs or Virtual Worlds for e-learning (Balladares, Menchaca, & Peredo, 2006; Monahan, McArdle, & Bertolotto, 2008). However, the communication modules of these applications are specific according to their environments and they cannot be used for any other systems.

Finally, another solution could be to combine the tools of CSCL environments with a VL. However, that way presents the important problem of making the user-interface more difficult and non-homogeneous.

The approach presented offers a new powerful tool for e-learning and remote teaching which enables any user (not developers) to make collaborative any VL developed with EJS. In contrast with above mentioned applications, the system proposed is very transparent and easy to use, and it can be also integrated in some CSCL or LMS.

3. Easy Java Simulations fundamentals

Easy Java Simulations is a freeware, open-source tool developed in Java, specifically designed for the creation of interactive dynamic simulations (Esquembre, 2004). EJS was developed for the Open Source Physics Project (Christian et al., 2003), which was established to create and distribute curricular material for physics computation. The computer simulations created with EJS can be used as stand-alone Java applications under different operation systems or be distributed via the Internet as applets. The principal reasons for choosing EJS as the platform to apply the collaborative system presented in this paper instead other software tools (Mathematica, MathCad, Maple, etc.), are the following:

- EJS have been designed for people who do not need complex programming skills. Teachers can easily and quickly create an interactive VL. Users need to provide only the most relevant core of the simulation's algorithm and the tool automatically generates all the Java code needed to create a complete interactive simulation, including a wide range of sophisticated software techniques (such as handling computer graphic routines, communication protocols, multi-threading and others). Besides, EJS also creates automatically an HTML page with the VL embedded in Java applet form.
- EJS has been conceived by teachers and for teachers, students who are more interested in understanding the simulated phenomena than in the underlying computer specific aspects.
- EJS is implemented in Java language. This feature gives a full portability of the VLs generated with this software and they can be executed by different platforms. In addition, Java language provides an extensive framework to develop Internet communications.

[2] Java Shared Data Toolkit: https://jsdt.dev.java.net/.
[3] Sun Developers Network: http://developers.sun.com/downloads.
[4] Java Sun downloads: http://java.sun.com/javase/downloads/index.jsp.

The process to create an application with EJS includes the definition of the model and the view (graphical interface). With regard to the model, the user must declare the variables which describe the system, must initialize them to their initial values and must write the differential equations that establish how these variables change in time or under user interaction. For this last step, EJS offers two options. The first is built-in editor of differential equations where users write the differential equations in a similar way to how they would write on a blackboard. EJS automatically generates the code that numerically solves the equations using one of the several standard algorithms (Euler, Euler–Richardson, Runge–Kutta, Runge–Kutta–Fehlberg, Dopri, etc.). This editor also includes support to handle simple state events of the differential equations. The second facility is a connection with Matlab/Simulink that lets users specify and solve their models with the help of these tools (Sanchez et al., 2005; Dormido, Farias, Sanchez, & Esquembre, 2005). Thus, the model can be defined with Matlab code in a Matlab function (m-file) or with a Simulink block diagram or with both. This flexibility allows VL's developers to create simulations for a wide range of physical or mathematical systems. In relation to the view, EJS provides a set of standard Java Swing, Java 2D and Java 3D components, and advanced graphical elements from the Open Source Physics Project (particles, vectors, images, vector and scalar fields) to build the interface in a simple drag-and-drop way. These graphical components have certain properties that the user can connect with the model variables and set a link between the model and the view. Therefore, the simulation turns into an interactive application where users can change the model variables and observe the simulation behavior in the view.

When a simulation is running, EJS creates and initializes the defined variables, and executes the differential equations in a differential time. This last phase is known as *step*. Later, the view is updated to show the new achieved state and executes again the equations to continue with the system evolution. During the execution of the simulation, EJS is able to get and control all the events that happen (from a user interaction o from the model). The user is who has to give functionality at these events. In order to do that, EJS provides a series of predefined functions to manage the simulation. The most important ones are the following:

- *Play*: the simulation is executed and begins to do *steps*.
- *Pause*: the simulation is paused.
- *Reset*: the simulation is stopped and the variables are updated with their initial values.
- *UpdateModel*: some model variables are updated.

In short, EJS can be considered a powerful tool for easily developing VLs. Up to now, there have been a lot of simulations developed with this software[5] for several science teaching fields (Engineering, Physics, Mathematics, Biology, Medicine, etc.). In addition, VLs developed with EJS are being integrated in CSCL environments as mentioned earlier (Gillet et al., 2005; Dormido et al., 2008).

4. System overview

One of the most important advantages of using Java technology to develop this system is that everybody can easily use it. As mentioned before, it only needs a web browser and the JRE (any version from 1.4 to 1.6) to create a collaborative virtual class with the presented approach.

In this section, a detailed system description will be explained. First, the main components which make up the collaborative system and the floor control will be described. Next, the communication protocols used in the real-time collaboration and how the applets are synchronized will be explained. Afterwards, the software architecture of the components is presented. Finally, some issues about latecomers, recording results and Java applet security will be shown.

4.1. Components and floor control

A fundamental issue in a synchronous collaborative system is the floor control (Dommel & Garica, 1997). This term points out how the system's components share the computational resources. The main objective of the collaborative system developed is to offer a shared VL that can be controlled in real-time by different members of a virtual class (students and teacher) and to be able to share the same experiments like in a traditional classroom. The shared VL is composed of an applet for each class member. All the Java applets have the same mathematical model and the same interface, and it is necessary to coordinate all them to achieve they will be synchronized. In this way, the actions of any member in the shared VL can be seen by all the rest of the members. Therefore, this e-learning system has two main components to coordinate: a teacher applet and some student applets. Initially, the teacher manages the VL controlling the simulation evolution in real-time. He/she has list of student simulations connected in the virtual session and he/she can disconnect any student at any moment. In order to have a suitable floor control and moderation, student simulations are locked and they cannot interact with the shared VL. They only are able to see what the teacher is doing in the shared simulation in real-time. Thus, the collaborative session avoids collisions among events which can cause unwanted and incoherent results. One example of this problem could be that the model of the VL becomes uncontrollable because of unsuitable user interactions.

Subsequently, in order to manage the virtual class like a traditional classroom, the system also has the feature of *chalk assignment*. With this option, the teacher gives permission to control the shared VL to a specific student connected, by selecting him from the list. The *chalk* only enables a student to manage the simulation, but not to manage the virtual session.

4.2. Communication framework

The communication framework of the collaborative system presented is based on TCP and UDP protocols. Student simulations are connected around the teacher applet in a peer-to-peer (P2P) centralized overlay network (Fig. 1). The teacher applet contains a multi-thread communication module which manages the synchronization of all VLs connected in the virtual class (see Section 4.3). In contrast with server-based approaches (Shirmohammadi et al., 1998; Shirmohammadi et al., 2003), this e-learning system is focused in a server-less archi-

[5] EJS's wiki: http://fem.um.es/EjsWiki.

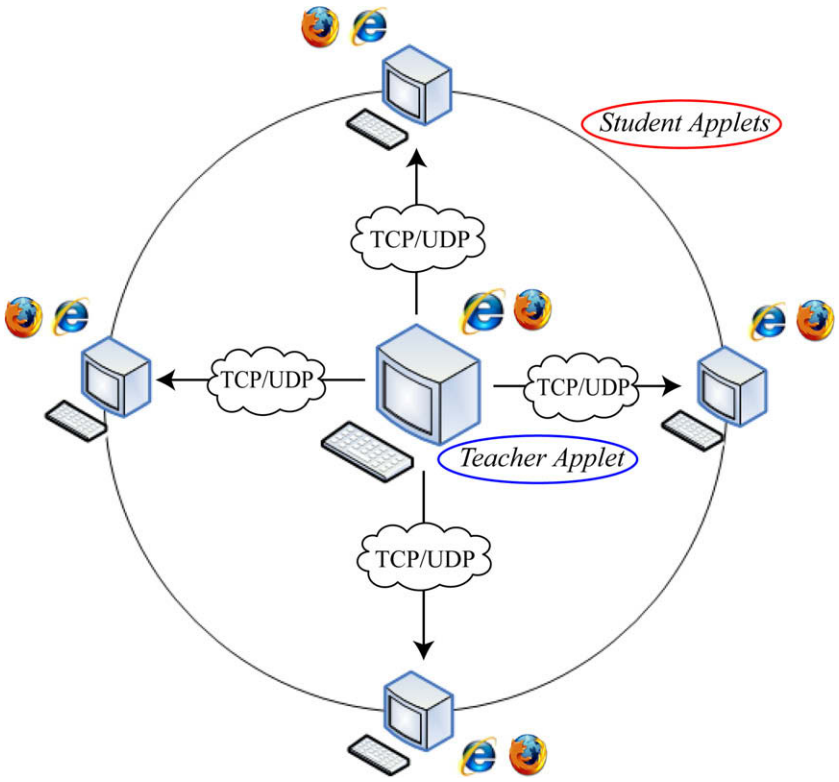

Fig. 1. Centralized overlay network performed in the collaborative system.

tecture. This communication method provides several advantages. First, it avoids the delay caused by the web server processing in the data flow. Second, users do not have to install any centralized server program in the web server because the communication engine is embedded in the Java applets downloaded. Third, the number of network connections can be substantially decreased. Fourth and finally, the teacher applet can manage the session, the floor control, and the data exchange. Thus, a centralized approach has higher control over student applets than other type of architectures.

As mentioned in the Section 2, Java applets can be synchronized in several ways. Most collaboration tools use a user event interaction framework (De Oliveira et al., 2003). However, this approach cannot be applied to VLs because some events may happen without an user interaction. Authors have developed a method based on *Java object tokens* (Shirmohammadi et al., 1998) which is able to synchronize in real-time all VLs connected to the virtual class. *Java object tokens* are small update messages which contain a string object which defines the action to perform. This approach sends commands through the Internet to VLs which are able to execute them because they are based on the same platform (EJS) and they have the same mathematical model. In addition, the small amount of information sent optimizes network bandwidth and reduces communication delays.

To synchronize the applets of the shared VL, the simulations must be in the same model state at all times. However, because of network limitations, update messages can be lost or delayed. Thus, authors have developed a communication framework which provides a transport service suitable for all update data: a TCP-based channel for reliable messages and an UDP-based channel for fast messages. Considering how EJS-based VLs work, the synchronization is organized as follows:

- Events such as *play*, *pause*, *reset* and *updateModel* are sent through a TCP channel. These important events need a reliability to synchronize the Java applets and this reliable protocol is suitable for that.
- Same state means that simulations must be in the same *step*. Therefore, every time the teacher applet executes a *step*, an object token is sent with the command "step" to connected student applets. These messages are sent through an UDP channel, faster but less reliable than TCP. Basically, UDP protocol is used for two reasons: because of the large amount of *steps* generated in the simulation (one per differential time) and the need to update quickly the student applets. In order to achieve a synchronization in real-time among all the applets connected in the virtual session, after sending a "step" command, the teacher applet pauses the simulation evolution and waits for "step done" response from student applets. After receiving this response from all students, the teacher applet continues with the system evolution. In order to not block the teacher simulation, a timer of 150 ms is invoked for every "step" command sent. If the timer expires before receiving all "step done" responses, the teacher applet sends an update of all the variables through the TCP socket (reliable connection). That way, all the applets of the shared VL are always synchronized.

4.3. Software architecture

The communication system explained in the previous subsection is embedded both teacher applet and student applets. Therefore, a user (teacher or student) is able to share experiences and to collaborate with the VL when the applet is downloaded. This subsection describes the teacher and student communication engine which are the software layer of the synchronized collaborative environment.

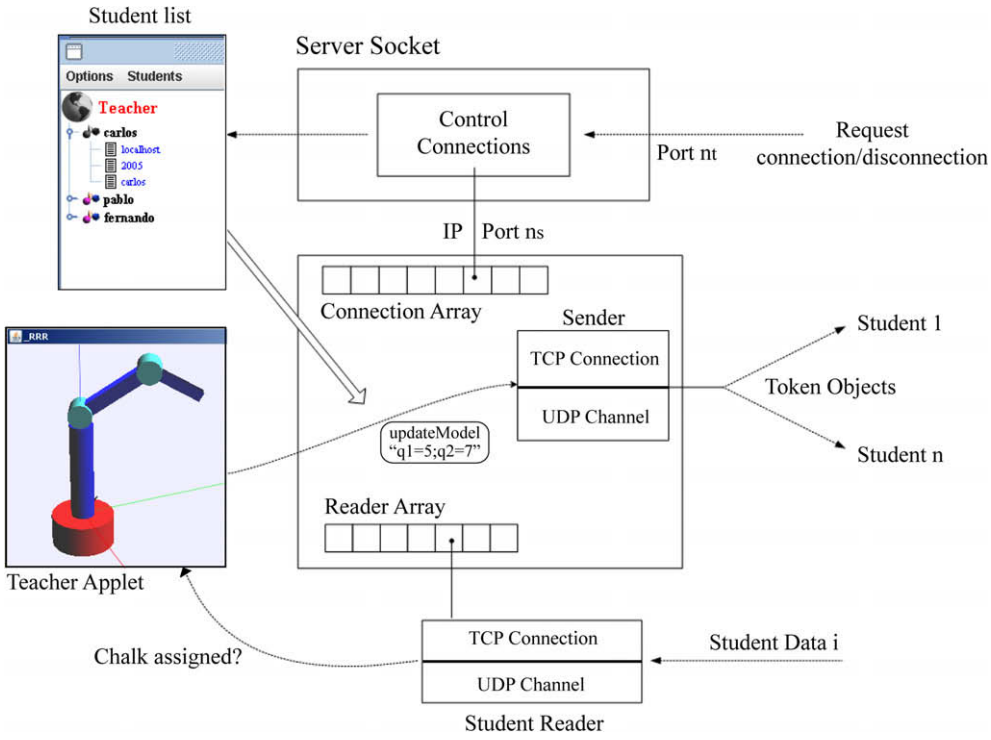

Fig. 2. Multi-thread teacher architecture.

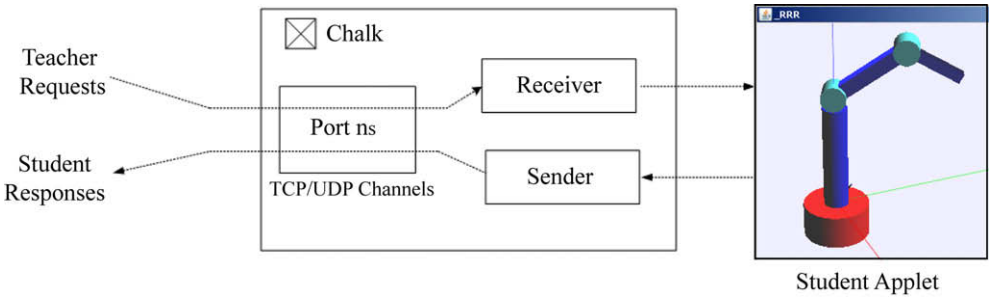

Fig. 3. Multi-thread student architecture.

4.3.1. Teacher architecture

Teacher applet is a multi-thread system that manages the synchronization among different student applets connected to the virtual class. The main parts which compose its communication system are shown in Fig. 2.

At the beginning of the virtual session, the teacher applet executes an instance of a *Socket Server* object which attends, new TCP requests from the applet students in a fixed port (*nt*). For each student connected, a new connection handler is added to the *Connection Array* and to the *Reader Array*. These objects contain two important data established by the TCP socket: the IP direction and the port opened in student applet (*ns*).

To synchronize the applets of the shared VL, the object *Sender* retransmits update messages to student applets connected (*Connection Array* elements). As mentioned in the previous subsection, these commands are embedded in *Java object tokens* and sent through the TCP and UDP channels. Each token is a small serializable Java object composed by a *String* label which is used to identify the message command and a *String* data object. For example, the corresponding *String* label of the event *updateModel* is simply "updateModel". The *String* data object represents a new simulation state with the variable's values changed, e.g. "$q_1 = 5$; $q_2 = 7$" (Fig. 2).

When a student has the *chalk* control, the teacher applet behaves like a student applet. Their controls are locked and it receives commands from the *chalk* student applet. The teacher communication system checks the *chalk assignment* and activates the corresponding reader from *Reader Array* to receive orders through the TCP and UDP channels (Fig. 2). Because of virtual class is a centralized approach, only the teacher applet receives update messages from the *chalk* student applet and retransmits them to the rest of student applets. *Step* synchronization is made in a similar way to that explained previously: the *chalk* student applet sends a "step" command to the teacher applet which retransmits this order to the rest of student applets connected.

4.3.2. Student architecture

Student software architecture is shown in Fig. 3. It is simpler than teacher architecture. The object *Receiver* is a thread which listens to update messages from the teacher through the TCP and UDP channels. It also manages "step responses" from the teacher applet when it has the *chalk* control. Finally, the object *Sender* is used to send "step responses" through UDP protocol and update messages in the *chalk* control mode.

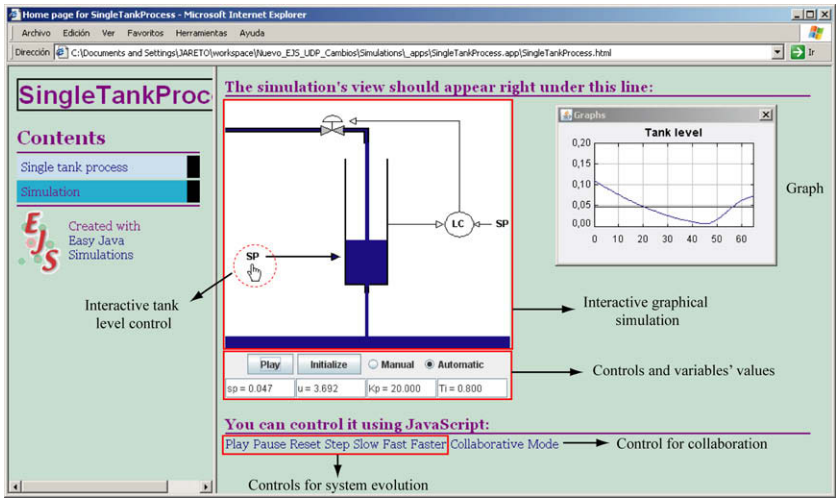

Fig. 4. Applet user interface.

4.4. Latecomers and recording results

When the teacher applet begins the collaborative virtual class, it considers that all student applets are connected. However, a student can join in the session already in progress. To synchronize this newcomer with the rest of applets, the teacher applet sends through the TCP connection (reliable connection) the current state of the VL. This token contains a *String* data object with the current value of all the model variables. Therefore, the latecomer is ready to receive update messages from the teacher applet and to take the collaborative virtual class in a synchronous way.

After finishing the collaborative class, students can use the VL individually. In order to have a play back of the teacher actions, student applets can execute the experiments performed in the virtual session. An experiment is known as the simulation execution in specific initial conditions (for example, some particular initial values for the parameters of a controller in a pumped tank process simulation). Therefore, these experiments can be used by students to revise the teacher virtual lesson and by latecomers to view all the virtual class. In addition, the teacher has a recording of the experiments performed by students when they interact with the shared VL in *chalk* mode.

4.5. Applet security issues

Applets downloaded through the Internet are prevented from reading and writing files on the client system and from making network socket connections except to the local host. Collaborative applets open socket connections to other applets through TCP and UDP protocols. Thus, shared applets must have the relevant permission whilst is executing in the client's PC.

An easy way to give permission to applets is using a *signed applet*[6]. The signature is a way to verify that the applet from a reliable source can be trusted to run in the computer.

5. How to use the collaborative system: A virtual class generation

This section describes how to create an e-learning collaborative environment with the approach presented in this paper. Basically, the process is composed of two points:

(1) The generation of a Java applet which is ready to teach in a virtual class (i.e., the teacher applet).
(2) The generation of a URL address for all the students in a public web server, where students can download their collaborative applets.

To show a more practical and realistic point of view, this section will be explained using a pumped tank process simulation (Vargas et al., 2006). The liquid level is controlled by means of a proportional-integral controller. Fig. 4 shows the different parts of the VL proposed, which has a very intuitive interface. As commented in Section 3, this HTML page with the VL embedded in Java applet form is generated automatically by EJS. Of course, this simulation is only a simple example of the power of EJS. Authors do not deem appropriate to describe the development of this VL because this is out of the scope of this paper and they recommend to consult EJS' references (Dormido et al., 2005; Esquembre, 2004; Sanchez et al., 2005).

5.1. Teacher applet generation

After developing the VL (model and view) with EJS, user only has to select the option "Generate Collaborative Applet" and complete two fields to generate a collaborative teacher applet. Both are defined inside options of the new EJS version (Fig. 5), and are the following:

[6] Security and Trust Services for Java: http://java.sun.com/products/satsa/.

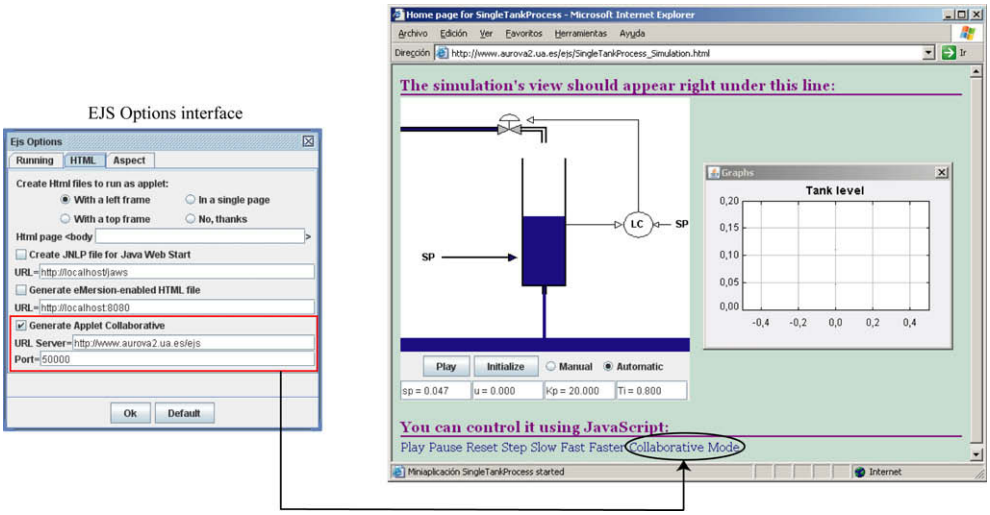

Fig. 5. Teacher applet generation.

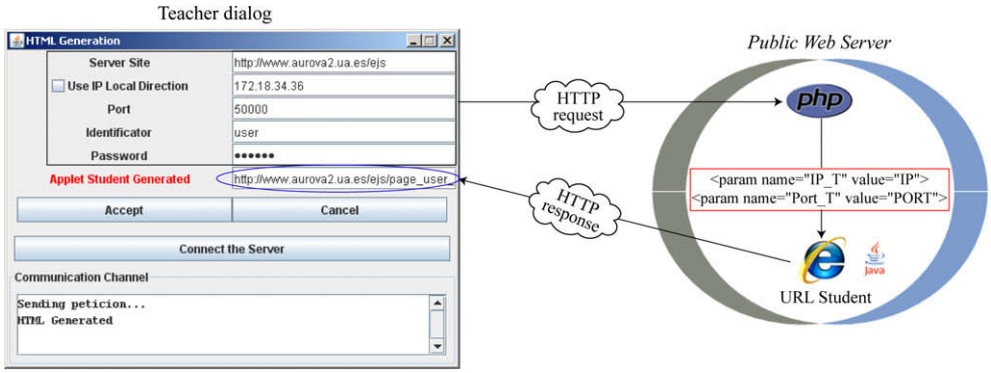

Fig. 6. Dynamic generation of a student URL.

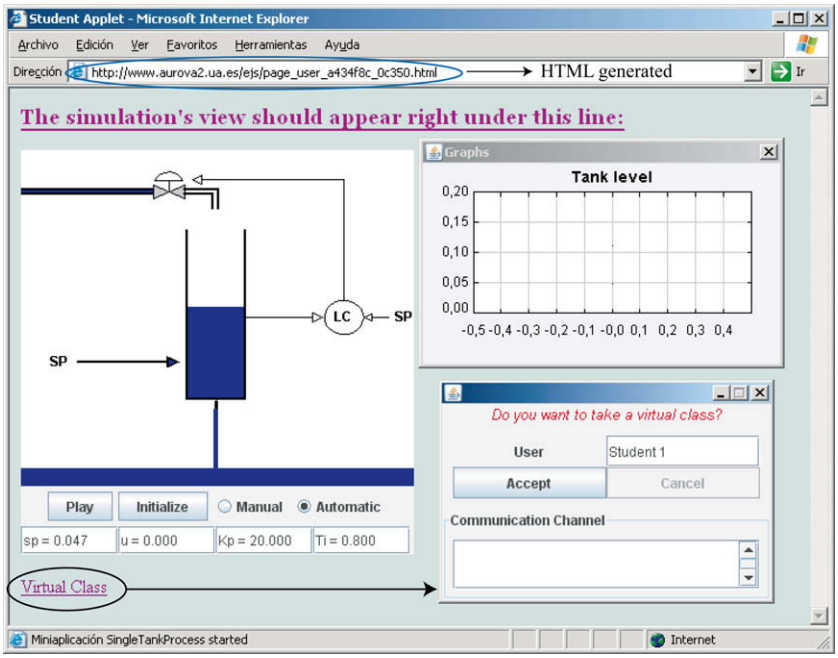

Fig. 7. Collaborative student applet.

- *URL server*: URL address of the public web server where the Java applets will be installed.
- *Port*: port number which will be opened for TCP and UDP communications in the virtual class. The user can utilize HTTP ports to avoid soft firewalls.

After filling these fields and running the simulation, a teacher applet prepared for collaboration is created. The generated applet is ready to be published in public a web server for sharing it. Fig. 5 shows the appearance of the VL proposed together with the EJS options. The control "collaborative mode" of the applet generated starts the collaborative system and generates the student applets in the public web server as it is described in the next subsection.

5.2. Dynamic student applet generation

In order to provide a URL address for students in a virtual class, a dynamic method to generate an HTML page in the public web server has been developed. This process is completely necessary to automatically provide the teacher's public IP address and the teacher's opened port for student applets. Thus, students will be able to take a virtual class in an easy and transparent way (i.e., they do not have to fill in any data about the teacher's IP and the teacher's port). The dynamic generation module is a simple PHP page installed in the public web server. To that end, only a public server with PHP 1.4 (o higher) support is required.

The student URL generation is performed directly from the teacher applet. The "collaborative mode" control executes a dialog called *HTML Generation* where some fields appear: *IP address*, where the teacher has to choose between local IP (for Intranet collaboration) or public IP (for Internet collaboration); *Identification* and *Password*, required for security; and *Server Site* and *Port*, which are the previous fields filled in EJS options. After filling them, the communication system of the teacher applet sends the dialog data via HTTP protocol. The PHP module located in the public web server receives this object data and generates an HTML file with the student applet embedded, ready to be accessed by students. The teacher's public IP and teacher's port are given to the student applet from the HTML generated (Fig. 6). The field *Applet Student Generated* will show the URL to be used by students to access to their collaborative applets. The teacher only has to send it to students by means of a simple collaborative auxiliary tool such as e-mail, forums or chat.

If the previous process is successful, the control "Connect the Server" will be enabled in the *HTML Generation* dialog. After activating this control, the teacher applet is listening to student applet requests on the port opened.

After that, students can connect to a collaborative virtual environment with only one URL address. They have to connect to the URL provided, activate the control "virtual class" and introduce a name in the dialog (Fig. 7). This is the identification for the teacher's student list.

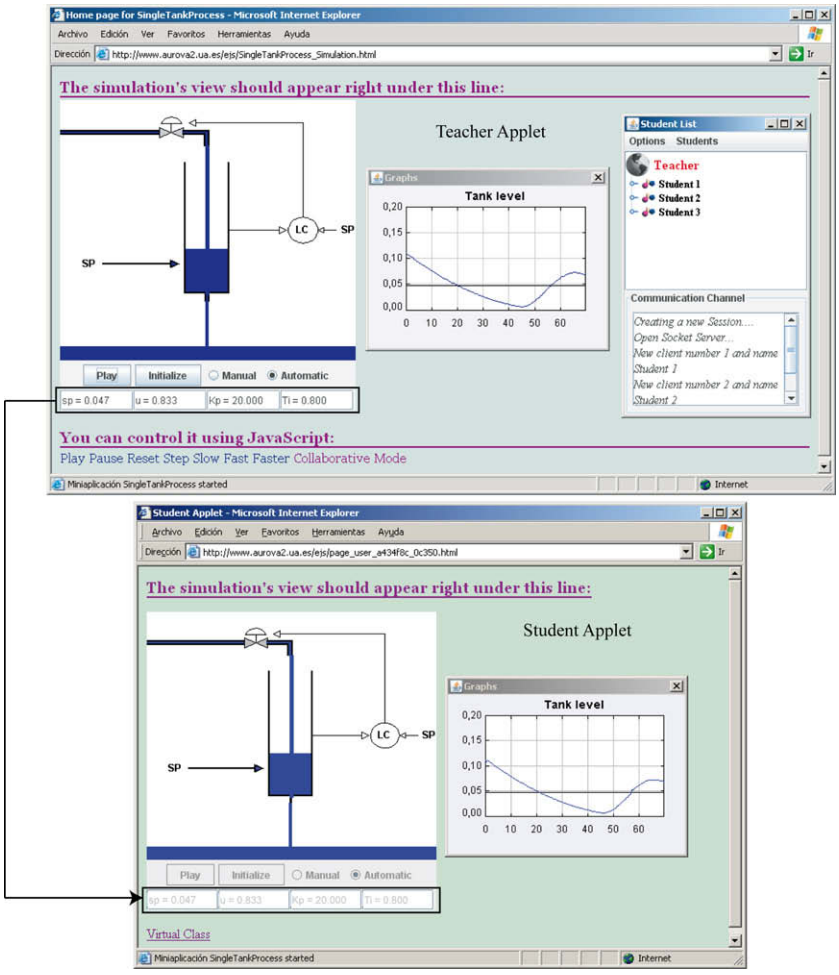

Fig. 8. Synchronized applets in a virtual class (I).

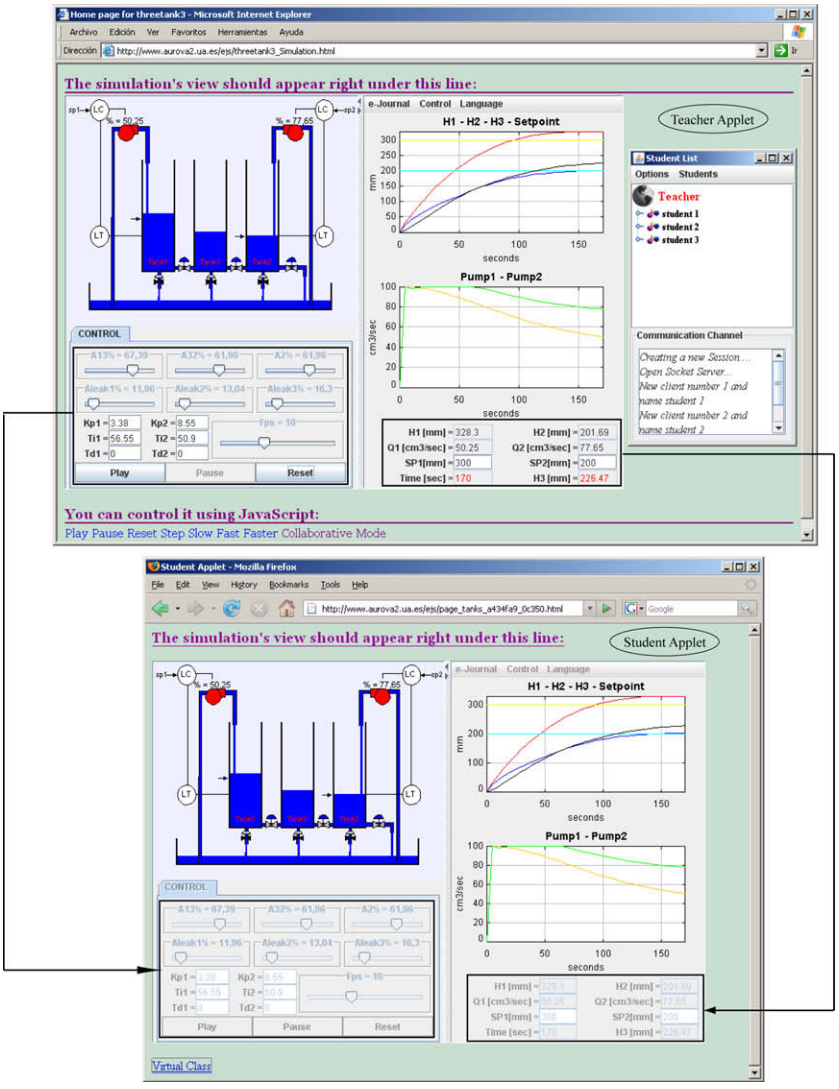

Fig. 9. Synchronized applets in a virtual class (II).

The communication engine of the student applet sends a TCP request to the remote teacher applet. Then, the student applet gets on-line with the teacher applet and joins in the collaborative virtual session.

In this way, teachers can create a collaborative environment from anywhere at anytime following the previously described steps. They only need an Internet connection and Java-enabled web browser to create a virtual class based on VLs.

It must be emphasized that teacher and student applets can work as individual simulations (i.e. without collaboration) while the options "collaborative mode" in the teacher applet, or "virtual class" in the student applet, are not activated. This allows students/teachers to practice VL experiments out of the on-line experience, as part of individual training.

5.3. Synchronization results

From the moment of the connection of the student applets in the collaborative session, their interfaces are disconnected. Student cannot experiment with the shared VL and the teacher applet manages in real-time the simulation evolution. Figs. 8 and 9 show the virtual sessions of two different VLs, one about the proposed VL (a single pumped tank process simulation) and another about a three tank system (Dormido et al., 2008). Each VL has three students connected. The teacher applet synchronizes all the student applets connected to the virtual session. In addition, the teacher can disconnect and give the *chalk* to any student from the student list at any moment.

As it can be seen in the Figs. 8 and 9, all the variables are synchronized (they have the same values). View and model are in the same state for all the VLs connected in the virtual class. Thus, student and teacher applets have the same variable values in real-time (field values emphasized in the figures). Tank level evolutions represent other real proofs about the synchronization of the VLs.

As commented in Section 4.2, after finishing the collaboration, students can test individually the experiments previously performed during the virtual class. They only have to press the right button over the VL and choose an experiment. Thus, they have a recorded version of the collaborative session.

6. Performance evaluation

Shared VLs are controllable environments which require real-time characteristics. Therefore, the presented communication approach must be able to support real-time applications within the acceptable parameters of human quality of service (QoS) for collaboration. This section shows an evaluation about the delay presented in a virtual class both in Ethernet/LAN and Internet networks.

As discussed in Section 4, the synchronization is achieved thanks to "step" orders. The teacher applet waits for "step done" responses from student applets to continue with the system evolution. The network delay caused by these responses can slow down the simulation and produce an unreal feeling in the system evolution. Although this problem is resolved (Section 4.2), a parameter that measures the quality of the collaborative framework is the average time from a "step" request to all the "step done" responses. This average time represents the time employed per *step* in a collaborative virtual session. From now, this parameter will be called *Collaborative Step Time* (CST).

As this approach is a TCP based centralized multi-user system (Fig. 1), the teacher applet has to send update messages to each student one by one and then wait the responses. Thus, CST increases with the number of students connected. The next two subsections studies the CST and how it changes in relation to student connections.

6.1. Intranet evaluation

The collaborative system was tested in a PC room with a LAN connection. The testing environment for collecting the performance results was set up with thirty Pentium IV-dual core PCs running at 3,4 GHz with 1GB of RAM. The PCs were connected via a 100 Mbps Fast Ethernet Switch. Windows XP with JRE 1.5 and two web browsers, Microsoft Explorer 7.0 and Mozilla Firefox 2.0, was running on all machines. In the collaborative experiment performed, both browsers were used with similar effect.

As shown in this paper, a virtual class consists of a teacher applet which manages some student applets connected. Thus, one of the computers was taken randomly from the PC room to be the teacher simulation. A teacher applet from the public web server was downloaded in this machine and it generated a URL address for students (Section 5.2). After that, the rest of computers were connected to the teacher applet by means of this URL.

The collaborative synchronous system was evaluated with a different number of student applets connected. Three different experiments were made to measure the CST: with 20, 40 and 60 students connected to the virtual session. As the PC room only had thirty machines, more than one web browser session was opened in the same computer to make the experiments. Fig. 10 shows the evolution of the average CST measured, and its maximum and minimum values in the experiment.

As expected, the CST increases with the number of student applets connected. As can be seen in the Fig. 10, the time delay evolution is nearly lineal. This is due to the fact that the teacher applet spends the same time processing per student connected. Analyzing the results, the small time delay measured in the three experiments does not produce any sense of blocking in the simulation and the synchronous system works correctly in LAN networks in terms of interaction and perception.

6.2. Internet evaluation

Internet performance was carried out from different places in the same country which had common Internet home connections. The Internet accesses for users ranged from a 1 Mbps/320 Kbps (download/upload) ADSL to a 5 Mbps/300 Kbps (download/upload) cable-modem. The system was evaluated with two, four and six students connected. Fig. 11 shows the average CST, maximum and minimum values measured in the Internet virtual session.

As expected, the CST was higher than in a LAN environment. The main reasons were a lower bandwidth (especially for uploads streams) and the distance to overcome among the connections. Another important reason of this delay was the loss of some UDP packets. It should be remembered that UDP protocol offers an unreliable stream and it is affected by the Internet traffic. The teacher applet can be delayed by the configured timeout during a simulation *step* because the loss of a UDP packet, although this does not block the simulation. Fig. 12 shows the UDP packet lost rate measured during the experiment for a virtual class of fifteen minutes long.

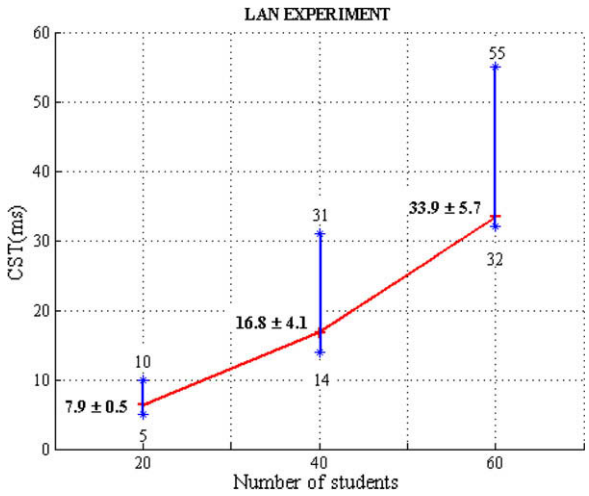

Fig. 10. CST results in a LAN network.

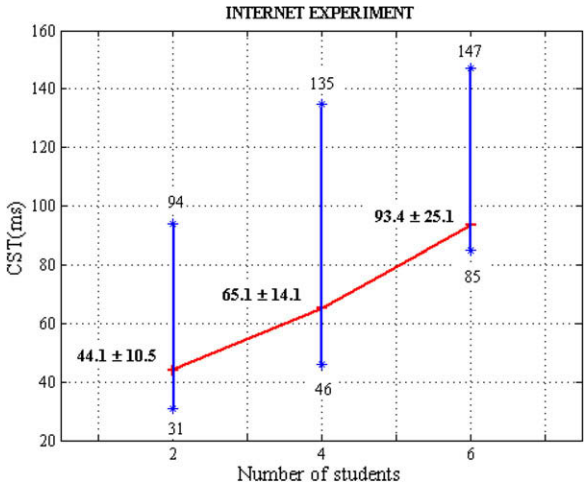

Fig. 11. CST results in the Internet experiment.

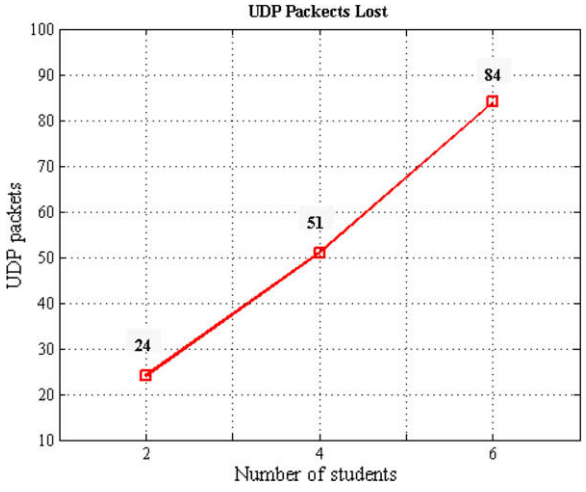

Fig. 12. UDP packets lost in the Internet experiment.

Despite the higher delays on Internet, the collaborative system developed can be considered a real-time application for human collaboration which offers the suitable QoS. Thus, this system can be used with other collaboration tools without any problem of synchronization.

7. Educational evaluation

At the moment, the collaborative system is being used in the courses *Computer Process Control* and *Robots and Sensorial Systems* in the Computer Science Engineering degree at the University of Alicante. The educational methodology is based on two points:

- Practical lessons at the University. The collaborative synchronous system is used in a PC room with a LAN connection. Teacher explains process control and robotic topics by means of VLs and students can follow the class on their own screen. In case of student doubts, the *chalk* mode is used. Thus, the student is able to manipulate the shared VL and to explain his/her doubt.
- Practical lessons through the Internet. Teachers are using this collaborative system with an on-line chat program to resolve student doubts from home. During the teacher tutorial time, students connect to the teacher from home and ask questions through the chat program. Teachers, by means of VLs and the collaborative synchronous system, resolve the student doubts.

In order to complete previous results about the need of collaboration in VLs (Candelas et al., 2003; Candelas et al., 2005), a new study was carried out in order to verify the usefulness and efficiency of the collaborative system presented. Questionnaire items were focused on three main issues:

(1) The suitability of VLs in the learning of relevant control and robotic concepts (I1).
(2) The functionality of the collaborative system (I2).
(3) The effectiveness of the synchronous collaboration in the learning process (I3).

Table 1 shows the questionnaire made to 25 students from the courses. Responses were rated on a five point Likert scale ranking from strongly agree (1) to strongly disagree (5). The statistical results obtained are reported in Table 2. Analyzing them, more than half of the

Table 1
Questionnaire to obtain user feedback

Question	Issue
In general, do you find easy the use of VLs for practical lessons?	I1
Did the VLs help you to understand theoretical concepts?	I1
Did you find it easy to do the practical exercises with the VLs?	I1
Did you find easy the use of the collaborative system?	I2
Did the collaborative system work correctly? Write the problems found	I2
Was the synchronization good in terms of perception and interaction?	I2
Was the synchronous collaboration among the VLs useful?	I3
Did you learn from the teacher's real-time feedback?	I3
Was the collaboration useful to share knowledge with your classmates?	I3

Table 2
Student questionnaire results in percentage of agreement per issue

Issue	Strongly agree (%)	Agree (%)	Neutral (%)	Disagree (%)	Strongly disagree (%)
I1	64	24	8	4	0
I2	52	24	0	20	4
I3	48	16	20	16	0

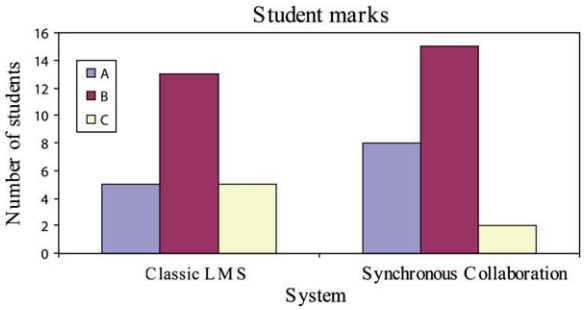

Fig. 13. Comparison of student marks.

students think that the VLs together with the synchronous collaborative system proposed are useful for distance practical lessons. In addition, most of the students consider the VL is an efficient tool in the learning process that encourages their ability to understand control and robotic concepts. However, some drawbacks were pointed out about the functionality of the collaborative system (issue I2). One of the most emphasized was related with the on-line chat program. They consider insufficient the use only of an on-line chat program in the collaborative environment to obtain a suitable teacher feedback. They suggested the use of other collaboration tools such as video or audio conference. For this, it could be a good idea to use current popular VoIP tools. Another piece of information reported was the appearance of problems in the communication from home using a cable-modem connection. They found some difficulties in the opening of the corresponding ports for the Internet collaboration.

In relation to the software to be installed, none of the students had problems with the installation of the necessary JRE. It should be emphasized that our students study Computer Science and they do not use to have problems regarding to new technologies issues. They adapt quickly to the user interfaces of the VLs. Moreover, they are able to resolve possible system startup problems by themselves. With students of other kind of non-technology-related studies could be necessary to organize an introductory seminar to use VLs. Nevertheless, the presented laboratories based on EJS have a very easy and intuitive user interface (Fig. 4).

With regard to the educational results obtained, both students who used a traditional asynchronous system (classic LMS) and students who used the new proposed system resolved the experiments more or less with a similar marks (Fig. 13). All the students passed the course. However, there is a few number of students who got better marks with the use of the collaboration system: 13% more with A qualification and 9% more with B qualification. The global qualification between two years increased from 7.39 to 7.82. In addition, for the first course year was needed a long tutorial process since it was used a traditional communication (emails and forums). In the second year, students learned the main concepts more quickly than the first year thanks to real-time sessions. In this way, students were able to resolve the experiments proposed before. Some students also were grateful because the synchronous system allowed them to resolve the experiments in a more guided way.

In addition, the collaborative system is going to be used in distance education among different universities from Spain. The universities involves are the following: University of Alicante, University Miguel Hernandez of Elche, University of Murcia and UNED ("Universidad Nacional de Educacion a Distancia"). They are getting ready for the next coming European Higher Education Area[7], in which the e-learning and the collaborative interaction will occupy very important roles (Lopez, 2005). These universities have important research activities in the field of virtual and remote laboratories, and they have been doing simulations with EJS for many years (Vargas et al., 2006; Duro et al., 2008; Buccieri, Sanchez, Dormido, Mullhaupt, & Bonvin, 2005; Guzman, Rodriguez, Berenguer, & Dormido, 2005). The Internet performed results

[7] The Bologna Declaration: http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf.

shown in Section 6 were obtained from a virtual class among these universities. Some of the simulations used by authors are available from the following web: http://www.aurova2.ua.es. These simulations are just ready to teach in a virtual class.

8. Conclusions

In this paper, a new web-learning system which combines two outstanding educational resources, the VLs and the synchronous collaborative learning practice, has been presented. With this approach, a new method to share knowledge in a synchronous way based on experiences over VLs has been achieved.

The use of VLs allows users to perform experiments from a remote location in an easy and practical way. They generate an environmental simulation which allows user interaction more intuitive and realistic. Moreover, it has been demonstrated that this web-learning resource is an efficient method to provide great hands-on experience to students. However, the presence of VLs inside collaborative environments is usually used in an asynchronous mode. This can make difficult the interaction among participants who have to do some experiments with the VL. In contrast, the approach presented offers a new tool to make collaborative synchronous environments which share VLs with only a Java-enabled web browser.

This e-learning system represents a portable collaboration framework for simulations developed in EJS. The communication architecture has been integrated in a new EJS version and can be applied in a transparent and easy way to all the VLs developed in this platform. Thanks to the collaborative features that the new EJS version includes into the applets, teachers can create a collaborative environment from anywhere at any time. They only need an Internet connection and Java-enabled web browser to access to the applets and create the virtual class. In addition, EJS makes easy the task of creating a computer simulation and provides a large amount of developed VLs.

Nowadays, many proposed on-line collaborative tools execute independent applications in the same computer's desktop, and this usually makes it difficult to use the interface. In comparison with these tools, this novel system provides an easy way to take a virtual class without using any external application. In addition, this approach can be used as a main tool for the CSCL synchronous environments for science teaching in areas such as Engineering, Physics, Mathematics and Biology in order to carry out practical lessons as well as to explain some theoretical concepts to a group of students. These scenarios are where the collaborative system has been successfully used by the authors.

Since the novel collaborative approach is based on Java applets (HTML pages), it can be included in the frame of on-line courses performed by LMS. Therefore, it is possible to apply any mechanism for learning management given by the chosen LMS. For example, authors are working in a module for Moodle to easily integrate a collaborative applet created with EJS into this LMS. In this way, all the course and student management is handled by Moodle, whereas the module supplies synchronous capabilities to the environment and makes possible use the collaborative system presented.

With regard to the results obtained in the evaluation, the collaborative system's behavior was successful in a LAN environment. The delay observed was minimal with a high number of students connected in the virtual session. Therefore, the system is able to manage a high number of students in a private LAN. Authors have carried out exercises in the faculty laboratory with a group of 60 students without performance problems. The experience gets a little worse when it was executed through the Internet. The delay measured in a virtual class of six students with a common Internet connection was lower than 150 ms in the worst case. Therefore, the collaborative synchronous framework developed is inside of acceptable parameters of the QoS. At present, the system works well with groups about 10 students, although it is not convenient manage great groups of students through the Internet. Authors are currently working on improving protocols to increase the performance in this last scenario.

Finally, authors are currently working on extending the collaborative synchronous system to apply it to remote laboratories. Thus, teacher and students will be able to share experiences in remote experiments using real equipment.

Acknowledgement

The work presented in this paper is supported by the "Ministerio de Educacion y Ciencia" of the Spanish Government through FPI Grants Program and Research Project DPI2005-06222. Authors would like to thank for this financial support.

References

Abdel, H. & Kvande, B. (1997). An Internet collaborative environment for sharing Java applications. In *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems* (pp. 112–117).

Abler, R., & Wells, I. (2005). Distributed engineering education: Evolution of the telecollaboration stations for individualized distance learning. *IEEE Transactions on Education, 48*(3), 490–496.

Alavi, M. (1994). Computer-mediated collaborative learning: An empirical evaluation. *MIS Quarterly, 18*, 159–174.

Bafoustou, G., & Mentzas, G. (2002). Review and functional classification of collaborative systems. *International Journal on Information Management, 22*, 281–305.

Balladares, L., Menchaca, R. & Peredo, V. (2006). 3D collaborative virtual environments over the web. In Z. Pan, R. Aylett, H. Diener, X. Jin, S. Göbel, & L. Li (Eds.), *Technologies for E-Learning and Digital Entertainment* (pp. 768–777).

Buccieri, D., Sanchez, J., Dormido, S., Mullhaupt, P. & Bonvin, D. (2005). Interactive 3D simulation of flat systems: The SpiderCrane as a case study. In *Proceedings of 44th IEEE European Conference on Decision and Control* (pp. 6222–6227).

Candelas, F., Torres, F., Ortiz, F., Gil, P., Pomares, J. & Puente, S. (2003). Teaching and learning robotics with Internet teleoperation. In *Proceedings of the Second International Conference on Multimedia and Information & Communication Technologies in Education* (Vol. 3, pp. 1827–1831).

Candelas, F., Puente, S., Torres, F., Segarra, V., & Navarrete, J. (2005). Flexible system for simulating and tele-operating robots through the internet. *Journal of Robotic Systems, 22*, 157–166.

Chabert, A., Grossman, E., Jackson, L., Pietrowiz, S., & Seguin, C. (1998). Java Object Sharing in Habanero. *Communications of the ACM, 41*(6), 69–76.

Christian, W. & Belloni, M. (2003). Developing open source programs for science and mathematics. In *Proceedings of the International Conference on Computer as a Tool (EUROCON)* (Vol. 1, pp. 15–19).

De Oliveira, J., Hosseini, J., Shirmohammadi, S., Malric, F., El Saddik, A., & Georganas, N. (2003). Java multimedia telecollaboration. *IEEE Multimedia, 10*, 18–29.

Dommel, H., & Garica, J. (1997). Floor control for multimedia conferencing and collaboration. *Multimedia Systems, 5*(1), 23–28.

Dormido, R., Vargas, H., Duro, N., Sanchez, S., Dormido, S., Farias, G., et al. (2008). Development of a web-based control laboratory for automation technicians: The three-tank system. *IEEE Transactions on Education, 51*, 35–44.

Dormido, S. (2004). Control learning: Present and future. *Annual Reviews in Control, 28*, 115–136.

Dormido, S., Dormido, R., Sanchez, J., & Duro, N. (2005). The role of interactivity in control learning. *International Journal of Engineering Education, 21*, 1122–1133.

Dormido, S., Farias, G., Sanchez, J. & Esquembre, F. (2005). Adding interactivity to existing Simulink models using Easy Java Simulations. In *Proceedings of 44th IEEE European Conference on Decision and Control* (pp. 4163–4168).
Duro, N., Dormido, R., Vargas, H., Dormido, S. Sanchez, J. Farias, G., et al. (2008). An integrated virtual and remote control lab: The three-tank system as a case study. *Computing in Science & Engineering, 10*(4), 50–59.
Esquembre, F. (2004). Easy Java Simulations: A software tool to create scientific simulations in Java. *Computer Physics Communications, 156*(2), 199–204.
Gillet, D., Anh, N., & Rekik, Y. (2005). Collaborative web-based experimentation in flexible engineering education. *IEEE Transactions on Education, 48*, 696–704.
Grundin, J. (1994). Computer-supported cooperative work: History and focus. *IEEE Computer, 27*, 19–26.
Guzman, J., Rodriguez, F., Berenguer, M., & Dormido, S. (2005). Laboratorio virtual para la ensenanza de control climático de invernaderos. *Revista Iberoamericana de Automática e Informática Industria, 2*(2), 82–92.
Hernandez, D., Bote, M., Asencio, J., Gomez, E., Villasclaras, E., Jorrin, I., et al. (1994). Free and open-source software for a course on network management: Authoring and enactment of scripts based on collaborative learning strategies. *IEEE Transactions on Education, 50*, 292–301.
Isenhour, P., Carroll, J., Neale, D., Rosson, M., & Dunlap, D. (2000). The virtual school: An integrated collaborative environment for the classroom. *Journal of Educational Technology & Society, 3*(3), 74–86.
Kamel, M., Taylor, A., & Breton, A. (2005). A synchronous communication experiment within an on-line distance learning program: a case study. *Telemedicine Journal and e-Health, 11*, 583–593.
Koschmann, T. (1994). Toward a theory of computer support for collaborative learning. *Journal of the Learning Sciences, 3*, 219–225.
Kreutz, R., Kiesow, S., & Spitzer, K. (2000). NetChat: Communication and collaboration via WWW. *Journal of Educational Technology & Society, 3*(3), 87–93.
Kuhmünch, C., Fuhrmann, T. & Schöppe, G. (1998). Java Teachware – The Java remote control tool and its applications. In *Proceedings of Educational Multimedia and Hypermedia World Conference* (pp. 70–75).
Lopez, P. (2005). e-Learning integration in our European higher education area. In *Proceedings of the Fourth International Conference on Multimedia and Information & Communication Technologies in Education* (pp. 412–415).
Ma, J., & Nickerson, J. (2006). Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Computing Surveys, 38*(3).
Marjanovic, O. (1999). Learning and teaching in a synchronous collaborative environment. *Journal of Computer Assisted Learning, 15*, 129–138.
Monahan, T., McArdle, G. & Bertolotto, M. (2008). Virtual reality for collaborative e-learning. *Computers & Education*, *52*, 1339–1353.
Moreno, L., Gonzalez, C., Castilla, I., Gonzalez, E., & Sigut, J. (2007). Applying a constructivist and collaborative methodological approach in engineering education. *Computers & Education, 49*, 891–915.
O'Malley, C. (1995). *Computer supported collaborative learning* (1st ed.). Berlin: Springer-Verlag.
Sanchez, J., Esquembre, F., Martin, C., Dormido, S., Dormido-Canto, S., Canto, R., et al. (2005). Easy Java Simulations: an open-source tool to develop interactive virtual laboratories using MATLAB/Simulink. *International Journal of Engineering Education, 21*(5), 789–813.
Shirmohammadi, S., De Oliveira, J., & Georganas, N. (1998). Applet-based telecollaboration: A network-centric approach. *IEEE Multimedia, 5*, 64–73.
Shirmohammadi, S., El Saddik, A., Georganas, N., & Steinmetz, R. (2003). JASMINE: A Java tool for multimedia collaboration on the Internet. *Multimedia Tools Applications, 19*(1), 5–28.
Vargas, H., Sanchez, J., Farias, G., Dormido, S., Dormido, R., Canto, S. & Esquembre, F. (2006). Web-based learning resources for automation technicians vocational training: Illustrated with a heat-flow and liquid level laboratories. In *Proceedings of the IFAC Symposium on Advances Control Education*.
Vicent, L., Anguera, J., Golobardes, E., Badia, D. & Segarra, M. (2005). Interactive multimedia contents and synchronous graphical communication tools for distance learning in Engineering Degrees. In *Proceedings of the 35th Frontiers in Education Annual Conference* (pp. F3G-5–F3G-6).
Wheeler, B. (2000). WebCT–WebCT clear leader in on-line learning programs. *The Chronicle of Higher Education, 11*, 34.
Yang, Z., & Liu, T. (2007). Research and development of web-based virtual on-line classroom. *Computers & Education, 48*, 171–184.