WEBSLIDE: A "Virtual" Slide Projector Based on World Wide Web[1]

Maria Barra,[2] Salvatore Ferrandino,[3] and Vittorio Scarano[4]

We present here the design key concepts of WEBSLIDE, a software project whose objective is to provide a simple, cheap and efficient solution for showing slides during lessons in computer labs. In fact, WEBSLIDE allows the video monitors of several client machines (the "STUDENTS") to be synchronously updated by the actions of a particular client machine, called the "INSTRUCTOR." The system is based on the World Wide Web and the software components of WEBSLIDE mainly consists in a WWW server, browsers and small CGI-BIN scripts. What makes WEBSLIDE particularly appealing for small educational institutions is that WEBSLIDE is built with "off the shelf" products: it does not involve using a specifically designed program but any Netscape browser, one of the most popular browsers available on the market, is sufficient. Another possible use is to use our system to implement "guided automatic tours" through several pages or Intranets internal news bulletins: the company Web server can broadcast to all employees relevant information on their browser.

KEY WORDS: Intranet applications; CGI programming; synchronous World Wide Web navigation.

INTRODUCTION

Teaching is a challenging activity: The instructor must communicate with the students in the best possible way. Often, it means to use technical devices to make the lesson more clear, more appealing, more enjoyable, that is, in a word (or rather two!) more fruitful.

Among the tools available to instructors, the projectors are, maybe, the more widely used: being able to show transparencies or (better) the video monitor of a Personal Computer is an important teaching aid. With the "Personal Computer Revolution" of the 80s, instructors and schools are actively using computers to prepare the so-called "courseware."

Unfortunately, often small schools and educational institutions can offer several computer labs to their students (maybe even a large number) but have fewer classrooms with projectors, if none at all.

Our system WEBSLIDE is a software solution to such a problem: how to use a computer lab also for teaching by providing a tool that allows to synchronously broadcast on the students video monitors the slides chosen by the instructor. The system does not require specialized hardware other than a (general-purpose) local area network connecting the PCs and a World Wide Web server and clients.

What we present here are the key concepts of the design of WEBSLIDE, which can reach only some of the capabilities shown in (Parnes *et al.*, 1996; Woo and Rees, 1994; Yeh *et al.*, 1996) but at (virtually) no cost: The design is simple, the mechanisms are available on a large subset of the browsers actually used, and the implementation is easy.

[1]A preliminary version of this paper has been presented at the 14th International Conference on Advanced Science and Technology (ICAST 98), Chicago, April 3–5, 1998.

[2]Dipartimento di Informatica ed Applicazioni "R. M. Capocelli," Università di Salerno, 84081 Baronissi (SA), Italy. E-mail: marbar@dia.unisa.it

[3]XCom Wide Communication, Via E.De Filippis 107/A—P.co Luciano, 84013 Cava De' Tirreni (SA), Italy. E-mail: salfer @xcom.it

[4]Dipartimento di Informatica ed Applicazioni, "R. M. Capocelli," Università. di Salerno, 84081 Baronissi (SA), Italy. E-mail: vitsca@dia.unisa.it

WEBSLIDE design offers the synchronization between instructor and students with a "minimalist" philosophy: Try to build a system that uses only "off-the-shelf" products that are currently available on the World Wide Web.

A prototype of WEBSLIDE has been realized in C and in PERL under the Unix operating system (for the server) using an Apache Web server and it has proven itself as a very efficient and cost-effective solution with low resources utilization.

Organization of the Paper. In the next section, we describe the World Wide Web and its application to the educational setting and then describe the HyperText Transfer Protocol (HTTP) and its limitations for the problem and briefly introduce some of the tools used by WEBSLIDE. Then, we show the key concepts of the design to such an extent that shows the reader the capabilities reached. We conclude the section (and the paper) with some extensions and future work.

THE WORLD WIDE WEB

The World Wide Web (WWW) (Berners-Lee, 1991); Berners-Lee *et al.*, 1992), developed in 1989 at CERN as a means of sharing information through the organization, has grown up as an information retrieval system on the Internet that can be considered the first real global hypermedia network. It was conceived as an environment where information from any source can be accessed in a consistent and simple way from any place in the network. Since 1989, WWW's growth has been exponentially fast, by reaching tens of millions of users through the whole Internet.

WWW and Education

Although originally developed to facilitate information sharing, the World Wide Web has large potentials as an educational support, especially for distance learning. In fact, several educational systems based on WWW have been developed in the recent past and the educational use of WWW has been a topic of several papers (Dwyer *et al.*, 1995; Ibrahim and Franklin, 1995).

In order to fully exploit WWW potentiality in the educational field, the characteristics of the model show some weaknesses. In fact, first of all, the stateless characteristic of HTTP makes impossible that server response is aware of user's behavior so that it can take into account the level of knowledge and, as a consequence, provide the user with documents that she can read, given her background/capacity. But, the major drawback for using the WWW as an educational system lies in the architecture itself: connections are always one-to-one (client-server) and sharing of information among group of clients is not allowed. This structure must be compared to what happens in "ordinary" classrooms: interaction is mainly between the teacher and students and the latter share what the teacher is saying or showing.

The aspect of the "ordinary" classroom that lies in the interactions between students and teacher and among students themselves has received a fair amount of attention. In fact, current research is exploring the potentialities of "Shared Workgroup" on WWW through several systems (Bentley and Horstmann, 1995; Chiu and Griffin, 1995; Gruber, 1995; MacArthur, Roscheisen, and Winograd, 1995; Virdhagriswaran *et al.*, 1995).

Far less attention, if compared with the previous one, has received the aspect of the navigation synchronization between the instructor and the students. Some encouraging results have been recently presented in this field in (Parnes *et al.*, 1996; Woo and Rees, 1994; Yeh *et al.*, 1996).

HTTP and State Information

HyperText Transfer Protocol (HTTP) is a very simple Internet protocol, similar to the File Transfer Protocol (FTP, RFC 959) and Network News Transfer Protocol (NNTP, RFC 977). HyperText Transfer Protocol is a fast and efficient protocol for searching and retrieving information from a server: the client makes a TCP-IP connection to the host by using domain name (or IP number) and the port number (default port is 80). Then the client sends a request document, consisting of lines (CR-LF terminated) of ASCII characters. The server sends back to the client its answer: the document required or (if that is the case) an error message.

Because of efficiency requirements, HTTP 1.0 was designed as a stateless protocol, that is the server does not keep any state on behalf of the client. In this way, HTTP servers can handle a large number of requests at the same time.

Although very efficient, this design requirement is limiting many possible activities that the server can perform on request. Examples are the ability of analyze the efficacy of cross references within the same Web site, build statistical profiles of users, to adapt its behavior on the basis of previous interactions and, last but not least, the ability to synchronize the behavior of several remote clients.

Solutions to these limitations fall in two categories: Those proposing an extension to the HTTP 1.0 protocol and those devoted to design a new protocol. In the first category, we see, for example, an extension to the HTTP 1.0 protocol (Kristol, 1996) which is a refinement of the "Cookie" solution proposed by Netscape (Netscape, 1995). On a different direction lies the effort to overcome this limitation in the next future by designing HTTP 1.1 (Fielding *et al.*, 1997), where it is allowed to use the same TCP/IP connection to perform multiple operations.

Both kind of solutions offer advantages but also suffer from some limitations. New HTTP 1.1 is still being designed and so are the browsers that can take advantage of the multiple operations capability.

On the other side, the "cookie" extensions proposed suffer from the drawback that they can be used on specific (also if widespread) browsers like Netscape 2.0 and following versions (Netscape, 1995). Also, widespread complaints over similar mechanisms that can be used to maintain surveillance of users accessing a Web server are reported over the Internet and taken into account by several authors (Hallam-Baker and D. Connolly, 1996).

CGI-BIN Program

Common Gateway Interface (CGI) is a standard for interaction between a WWW server and external application. A Common Gateway Interface program (often called "CGI-BIN script") is an executable program residing on the server whose execution can be triggered by a remote client. CGI-BIN scripts get input data from the client either by encoding data in the URL (Uniform Resource Locator) (often said *URL-encoded*) as a string prefixed by '?' and separated by '+' or (when input data can be very large, for example) On standard input. In the latter case, the client does an HTTP POST or a PUT operation and the data is attached to the request itself. Output is sent back to the remote client, and, consequently, it is often in HTML format.

Netscape Server Push—Client Pull

In an effort to mitigate the lack of dynamicity that is currently offered by HTTP protocol and by the structure of clients, Netscape has developed and integrated in their browser two mechanisms that can be used to implement dynamic documents.

The characteristics of the two mechanisms, "client pull" and "server push," are described in Netscape (1996). The "client pull" mechanism is when the server sends information (embedded in a HTML page) that indicates the client that it should "load" something (possibly the same document) within a specified number of seconds. A new connection between the client and (possibly the same) server is, then, opened and the action performed. This is done by using a <META> tag in HTML which allows to specify information to be included in the HTTP header. In this case, a line in the file such as <META HTTP-EQUIV="Refresh" CONTENT =10;> asks the client to go ask again the same URL at the server in 10 seconds. It is also specify a different URL to be fetched at the deadline.

On the opposite side lies the other mechanism: the "server push." In response to a GET request, the server sends its data enclosed in a experimental MIME type/subtype, called multipart/x-mixed-replace, that allow to send several documents "at once."

Documents are separated by "boundaries" (i.e., strings that do not appear in the document being sent, prefixed by double hyphen) as specified in "standard" multipart Mime types (Borenstein and Freed, 1993). There are part boundaries that enclose an object that can be treated by the browser and a final boundary that indicates that the transmission is over. When the first document is completely received by the browser (i.e., when the browser receives a part boundary), it shows the document but, in the meantime, it is receiving the second document. When the second document is also completely received, then it replaces the previous one on the browser, and so on. The TCP connection between client and server is kept open by the server and the client is waiting on the other side, that either new document are transmitted or the final boundary is sent by the server. The connection may, then, be kept always open by a server that wants to be able to send and update "synchronously" a document to a client. The server push mechanism has been used as a simple way to obtain animation: The server push can, in fact, be used for pushing images in a <IMG> tag.

Our idea is mainly to use server push (and to a lesser extent, client pull) to obtain a synchronous behavior between several clients and a server.

THE WEBSLIDE

In this section we describe WEBSLIDE'S structure and show how it can be easily implemented. Finally, some possible scenario where WEBSLIDE can be useful are shown.

The Principle

WEBSLIDE is built on some assumptions: First, CGI-BIN scripts are necessary to allow communications between clients connected at the same server. Then, clients have to recognize documents that are "server pushed" through the same TCP connection. Third, some interprocess communication is available on the server.

In the sequel, we call INSTRUCTOR and STUDENTS several clients connected at the same server: the behavior of WEBSLIDE is such to guarantee that local links followed by the INSTRUCTOR are shown on STUDENTS browser.[5] Then, the synchronization process can be described in several phases defined as follows:

1. Setting up the classroom. The INSTRUCTOR sets a "class" by providing a topic and by specifying other optional information such as, for example, setting a maximum number of STUDENTS that can be "attending" the lecture or setting the start time.
2. Enrollment of STUDENTS. Each STUDENT can check the available classes (through a particular page available on the server) and enroll in one of the classes. It is also given the chance of getting the list of attendants.
3. The lesson. The INSTRUCTOR starts the lesson by following links through previously prepared slides in HTML. Each page that is shown on her browser is synchronously shown on STUDENTS' screen and she can interact with her STUDENTS through audio-live connections or through more conventional means.
4. End of the lesson. The INSTRUCTOR follows an "end of the lesson" link that closes the TCP connections toward STUDENTS.

[5]Our distinction among clients is similar to the *master/slave* clients in Yeh *et al.*, 1996.

The Structure

We describe here WEBSLIDE'S structure by showing the actions that must be performed at each of the phases previously described and how that can be easily accomplished by using several intercommunicating CGI-BIN scripts.

State Info Mechanism

First of all, let us notice that a small amount of information has to be exchanged between the server and the INSTRUCTOR and between the server and the STUDENTS. In fact, for example, when the INSTRUCTOR sets up the classroom it must "receive" the Class Identifier (CID), that has to be used in every future interaction with the server. Given the stateless property of HTTP the server "memory" is explicitly implemented[6] through the "parameter passing" mechanism (as in Ferrandino *et al.*, 1996).

This solution needs a simple CGI-BIN script which takes as input (URL-encoded) a filename and some "state" information. The script, when is executed, returns the required document taking care to substitute any "internal" link (within a certain realm, that can also be the server itself) to a file called foo.html with a call to the script with first parameter foo.html and the instance of the state as it is known by the script itself at that time as other parameters. In Ferrandino *et al.* (1996), the mechanism is used to implement adaptive response from an HTTP server depending upon previous interactions with the same user. Here, the mechanism can be used two times:

- When the INSTRUCTOR sets up the classroom, it needs the CID so that any future interaction with its STUDENTS can be managed by the server. In this case, the CGI-BIN script modifies the links in such a way to "embed" the CID as second parameter.

[6]Given that we are using Netscape browser, it can also be implemented by using the well-known "Cookies" mechanism (Netscape, 1995), but some criticism raised in (Hallam-Baker and D. Connolly, 1996) about privacy issues suggested us to choose the explicit mechanism.

- When the STUDENT asks for a list of available classes, she needs to follow a link that is a CGI-BIN script with parameter the CIDs for every available class. When the STUDENT chooses the class, she is enrolled into the class shown by the CID.

The Phases

Let us describe how each phase can be implemented by using "off the shelf" software products. The idea is as follows: the server has several processes (one for each student) that take care to push toward the STUDENTS the page as soon as the INSTRUCTOR requests it. These processes keep the TCP connection open and, at the end of each file, send a "part boundary" to signal the STUDENTS that it can be shown and more is coming.

First of all, let us see what has to be done in the first phase, when the INSTRUCTOR sets up the classroom. The INSTRUCTOR needs to interact with the server to establish if there is "enough room" for her classroom. In fact, this phase could be critical since the WEBSLIDE mechanism is based on spawning a certain number of child processes that could easily overload the server. Then it is highly suggested that access to the first phase is provided with some security measure (a simple password mechanism might be enough).

The (would-be) INSTRUCTOR fills up a form where its requests for a class are taken (title of the class, maximum attendance, first page to be shown, etc.) and, in response, the server returns a starting document with the information about CID embedded into the local links with calls to a CGI-BIN script, called MONITOR. This script takes care of receiving the requests from the INSTRUCTOR and communicate with processes that take care of the STUDENTS (more details follow when we describe Phase 3). The MONITOR also takes care of inserting into the document a link to the "End of the lesson" which is a call to the MONITOR itself with null first parameters.

Moreover, the server takes care of creating a data structure (either in shared memory or on the disk) for the enrollments (the "room" for the STUDENTS) and also to update a page of "available classes" for consultation. A scheme of the actions taken in Phase 1 is given in Fig. 1.

In Phase 2, each (would-be) STUDENT asks the server the "available classes page" where each link, if followed, is a call to a script, called the TUTOR, that takes care of the student enrollment and of all the future interactions with the class. First, the TUTOR updates the data structure used to represent the "room" by writing its CID and then send back a "Please Wait" page starting a "server push" on the STUDENT. At this point, being enrolled in that class, when the lesson starts, the TUTOR will be notified by the MONITOR (executed by the INSTRUCTOR) that it has to read a given page and send it back to the STUDENT it is in charge of. A scheme of the actions taken in phases 2 is given in Fig. 2.

Now, let us see what happens during Phase 3 (the scheme is available on Fig. 3). The INSTRUCTOR chooses links as desired. Links were changed in phase 2 so that they activate the MONITOR that takes two steps:

- First, it sends back the page required, taking care of changing the internal links in such a way to bring the CID as a parameter.
- Then, for each STUDENT it communicates with its TUTOR by sending the name of the file to be pushed toward its STUDENT. The TUTOR takes also care to include a "part boundary" of the multipart/x-mixed-replace MIME type so that the STUDENT browser will show the page as soon as it is completed, while keeping the TCP connection open.

Phase 4 is much simpler: If the MONITOR is called with a null first parameter then it knows that it is the end of the lesson, and acts as a consequence. The TUTORS are asked to send the "end boundary," possibly after a "Come back soon" page to each student, and the MONITOR is also in charge of "cleaning up" the room (deleting shared memory or files, updating the list of available classes, etc.).

Applications and Extensions

We examine here, a few other applications when WEBSLIDE can be useful. First of all, it can be used also like automatic slide changing on several monitors that are in a museum or similar. In fact, the system can be made "automatically changing" slides: It is enough to use a client-pull mechanism: the INSTRUCTOR, in this case, is required only

INSTRUCTOR	Server
Asks the server for allocation of a "room". ↘	
	Allocates the "room" (if available) and returns a starting page with Class ID (CID) embedded within the links. ↙
Waits for the start time.	

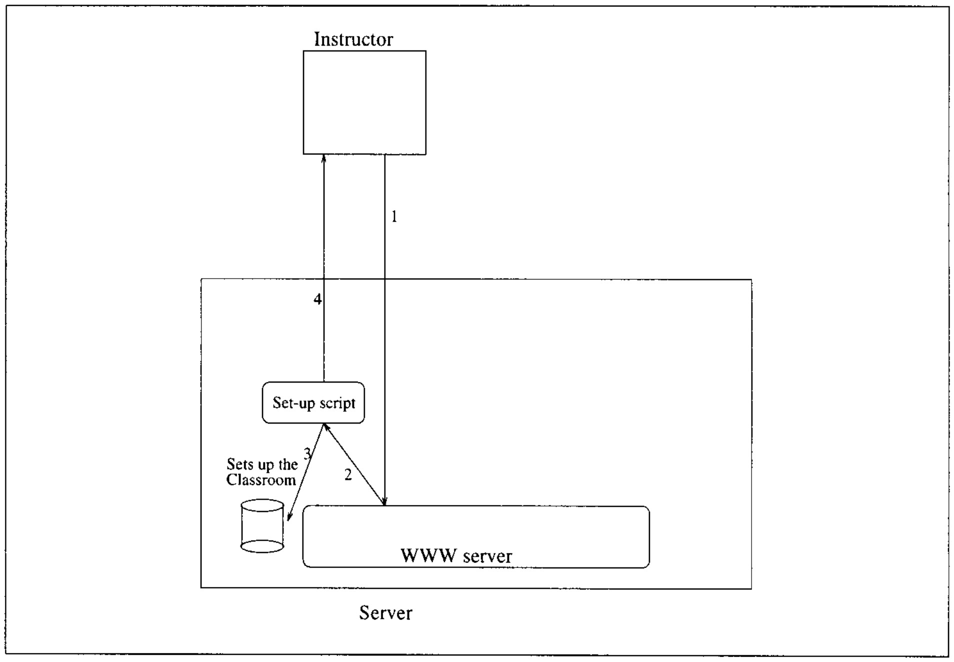

Fig. 1. The scheme for Phase 1 (Setting up the classroom) and the corresponding action diagram.

to start a sequence of pages that are circularly linked via a <META HTTP-EQUIV="Refresh" CONTENT=10; URL=http://foo.bar/pageyy.html> HTML tag. The STUDENTS will simply follow the flow of pages on their screen. The INSTRUCTOR can also insert at "run-time" new pages or update the old ones.

Such tours have been very useful in avoiding any cognitive overhead and disorientation. Work in this area goes back to almost 30 years (as Bush's trails (Bush, 1967)) but also to recent papers (see Trigg's guided tours (Trigg, 1998) and "Footstep" mechanism (Nicol *et al.*, 1995).

Another useful application could be as an information broadcast on an Intranet: think of all the employees having a browser open on the "What's new today" information of the company they are working with. Updated information is sent immediately to all the employees.

Some extensions are possible in the design and can be easily included in the prototype. For example, it is possible to allocate a classroom "off-line" and not require the teacher to wait "on-line" for the lesson to start. It can be done by allowing to create a CID and then when the INSTRUCTOR follows a password-protected link the lesson starts.

Server	STUDENT
	A student asks for available classes to the server. ↙
Server sends back a page with links to starting points and CIDs embedded in the links. ↘	
	A student follows one of the links and enrolls herself into the class. ↙
Server starts a "server-push" process (the TUTOR) that communicates with the student and "enrolls" the student in the class by taking note of TUTOR's PID for that student. ↘	
. . . *repeated for all* STUDENT*s.*	

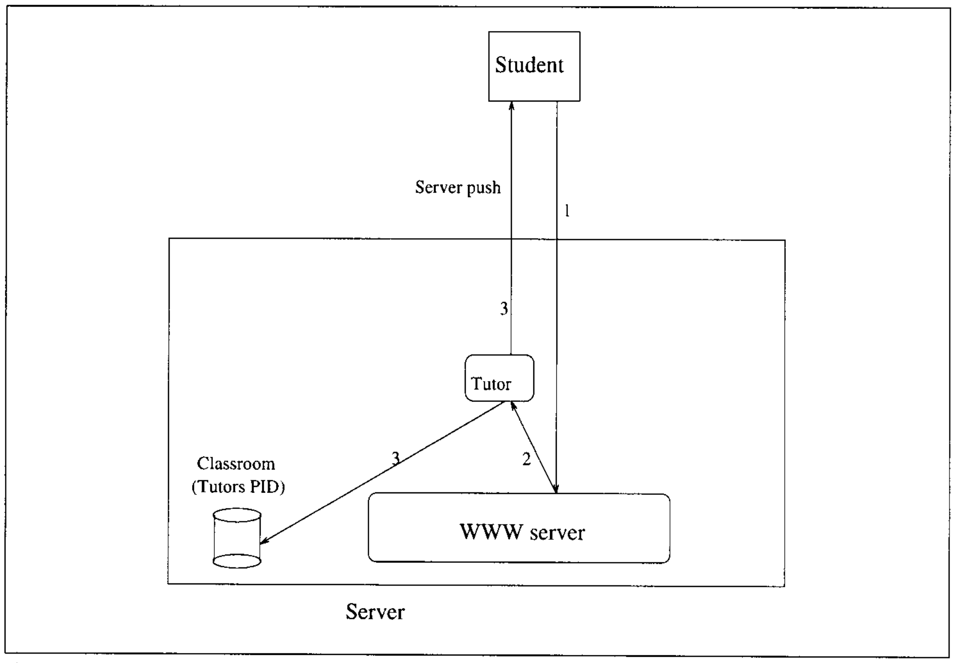

Fig. 2. The scheme for Phase 2 (Enrollment of STUDENTS) and the corresponding action diagram.

Another important extension, that is planned next, extends the WEBSLIDE to the whole WWW. In the current design, the TUTORS receive the name of the file to be sent that (in our assumptions) is assumed to be local to the server. In a different scenario, the server could take care to act as a proxy by downloading remote documents (as required by the INSTRUCTOR) and then sending its filename to the TUTORS. This could easily increase the effectiveness of a lesson: the teacher can include HTML pages everywhere in the Web.

REFERENCES

Bentley, R., and Horstmann, T. (1995). *Supporting collaborative information sharing with the World Wide Web: The BSCW Shared Workspace system.* Workshop on World Wide Web and Col-

INSTRUCTOR	Server	STUDENT
Follows one of the links (probably something like "Next" link) which "brings" embedded information for the server like the CID. ↘		
	Returns the required document, modifying its links so that they bring embedded the CID. ↙	
Gets the file that is going to be shown to STUDENTs	Writes the filename of document to a shared memory or file and make the TUTORs send the document to the whole class. ↘	
		Each student receives the new document that is shown on the screen, while the TCP connection is left open.
... repeated for all links followed by the INSTRUCTOR.		

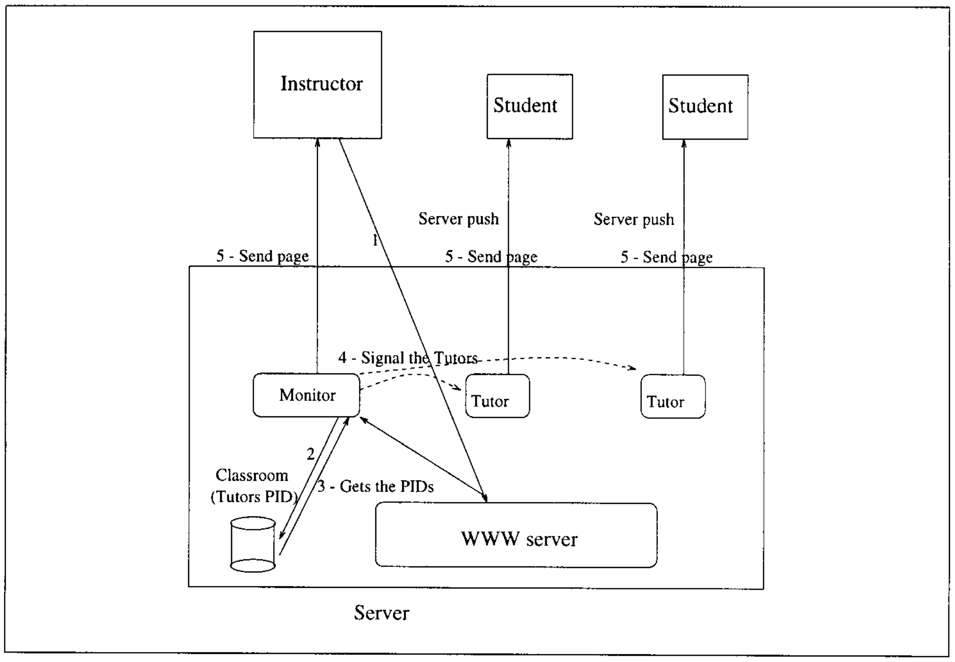

Fig. 3. The scheme for Phase 3 (the lesson) and the corresponding action diagram.

laboration, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Berners-Lee, T. (1991). *World Wide Web Initiative.* WWW Home Page. [http://info.cern.ch/hypertext/WWW/TheProject.html]

Berners-Lee, T., Cailliau, R., and Groff, J. F. (1992). The world wide web, *Computer Networks and ISDN Systems*, 25: 454–459.

Borenstein, N., and Freed, N. (1993). *MIME (Multipurpose Internet Mail Extensions). Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies,* RFC 1521, Sept. 1993.

Bush, V. (1945). As we may think, *The Atlantic Monthly*, July.

Chiu, D. M., and Griffin, D. (1995). *Workgroup Web Forum: Tools and Applications for WWW-Based Group Collaboration.* Workshop on World Wide Web and Collaboration, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Dwyer, D., Barbieri, K., and Doerr, H. M. (1995). *Creating a Virtual Classroom for Interactive Education on the Web.* Proc. of WWW 95, Third International Conference on World Wide Web.

Ferrandino, S., Negro, A., and Scarano, V. (1996). *CHEOPS: Adaptive Hypermedia on World Wide Web.* Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Telecommunicazion Services (IDMS '97), 10–12 Sett. 1997. Ed. Springer-Verlag (LNCS).

Hallam-Baker, P. M., and Connolly, D. (1996). *Session Identification URI.* W3C Working Draft WD-session-id-960221. [http://www.w3.org/pub/WWW/TR/WD-session-id.html]

Fielding, R., Frystyck, H., and Berners-Lee, T. (1997). *Hypertext Transfer Protocol,* HTTP 1.1. HTTP Working Group Internet Draft.

Gruber, T. (1995). *Collaborating around Shared Content on the WWW.* Workshop on World Wide Web and Collaboration, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Ibrahim, B., and Franklin, S. D. (1995). *Advanced Educational Uses of the World Wide Web.* Proc. of WWW95, 3rd International Conference on World Wide Web.

Kristol, D. K. (1996). *Proposed HTTP State Management Mechanism.* HTTP Working Group Internet Draft (2/22/96).

LaLiberte, D., and Bravermann, A. (1995). *A Protocol for Scalable Public Group and Public Annotations.* Workshop on World Wide Web and Collaboration, Massachusetts Institute of Technology, Cambridge, Massachusetts.

MacArthur, K. *Collaboration, Knowledge representation and Automatability.* [http://www.w3. org/pub/WWW/Collaboration]

Netscape Communications Corporation. (1995). *Persistent Client State HTTP Cookies.*

Netscape Communications Corporation. (1996). *An Exploration of Dynamic Documents.*

NCSA Mosaic Project (1994). *NCSA Mosaic Home Page.* [http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html]

Nicol, D., Smeaton, C., and Falconer Slate, A. (1995). Footsteps: Trail-Blazing the Web. Proc. of WWW 95, Third Internazional Conference on World Wide Web. [http://www.igd.fhg.de/www/www95/proceedings/papers/60/footsteps.html]

Nielsen, J. (1990). *Hypertext and Hypermedia.* Academic Press Ltd.

Parnes, P., Shefström, S., and Synnes, K. (1996). *WebDesk: The Tele-Conferencing Services of MATES.* ERCIM Workshop on CSCW and the Web, Feb. 1996. [http://mates.cdt.luth.se/papers/SMAC/Webdesk_SMAC.html]

Roscheisen, M., and Winograd, T. (1995). *Generalized Annotations for Shared Commenting, Content Rating, and Other Collaborative Usage.* Workshop on World Wide Web and Collaboration, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Trigg, R. H. (1988). *Guided Tours and tabletops: Tools for communicating in a hypertext environment.* ACM Transactions on Office Information Systems 6,4 (October).

Virdhagriswaran, S., Webb, M., and Mallatt, J. (1995). *Shared Information Space: An Interactive, Collaborative System Enablement Perspective.* Workshop on World Wide Web and Collaboration, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Woo, T. K., and Rees, W. J. (1994). *A Synchronous Collaboration Tool for World-Wide Web.* Proc. of the 2nd World Wide Conference [http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/CSCW/rees/SynColTol.html]

Yeh, P., Chen, B., Lai, M., and Yuan, S. (1996). *Synchronous Navigation Control for Distance Learning on the Web.* Proc. of 5th International World Wide Web Conference.