International Jl. of Educational Telecommunications (2001) **7**(2), 157-194

Towards a Telecommunication Platform for Supporting Distributed Virtual Laboratories

SAMUEL PIERRE AND MARTHE KASSOUF
Mobile Computing and Networking Research Laboratory
Department of Electrical and Computer Engineering
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-Ville
Montréal, Québec, Canada, H3C 3A7
samuel.pierre@polymtl.ca

This article presents a telecommunication platform dedicated to supporting distributed virtual laboratories. The architecture of this platform is made of three layers. The first layer deals with interoperability among heterogeneous networks and allows users to access virtual laboratory environments. The second layer provides a set of tools and generic functionalities sharable among several specific laboratories. The third layer insures the adaptation of these basic tools to other specific tools that exist in the peculiar context of each laboratory. The result is a distributed environment in which diverse elements interoperate to provide not only for pedagogical contents but also access mechanisms to simulations and virtual experimentation from different specific laboratories. The experimentation results put the network congestion level and the quality of service required by the user on top of the criteria determining the global efficiency of this platform.

The more computer networks evolve, the more the variety of machines related to them and the links they use increase. In fact, each type of network has its own specific logical setting, mode of switching, data format, and level of quality of service. This explains, in part, the existence of heterogeneous environments for public and private networks of boundless

dimensions giving rise to many problems of incompatibility and interoperability (Prnjat and Sacks, 1999).

This article proposes a telecommunication platform model that insures interoperability among heterogeneous networks and serves as an access infrastructure to distributed virtual laboratories. Conceptually, this platform is a three-layered structure where a layer of basic tools and functionalities (BTF) is framed with an adaptation layer that adapts them to specific tools and functionalities (STF).

The main idea consists of modeling a generic laboratory node whose additional extensions will enable development of other specific laboratories for diverse scientific and engineering disciplines. To start, two laboratory prototypes, one in physics and another in electrical engineering were developed. Then, other specific prototypes for chemistry, biology, computer engineering, and mechanical engineering were added. The envisioned generic model integrates a more complete set of external possible attributes common to diverse specific laboratories. These external attributes constitute the basis for some basic tools and functions shared by all specific laboratories. Each laboratory is given specific characteristics and functions referred to as specific tools and functions.

The design approach adopted was both object-oriented and inductive. It was object-oriented because specific laboratories inherited a number of characteristics from a generic laboratory that became a common ancestor. It is inductive, because the enrichment of the generic model resulted in incremental adding of diverse groups of internal attributes generated from the future development of a sufficient number of specific laboratories. The following is the list of acronyms used in this article:

ABR: Available Bit Rate
ADSL: Asymmetric Digital Subscriber Line
API: Application Programming Interface
ATM: Asynchronous Transfer Mode
BSD: Berkeley Standard Distribution
BTF: Basic Tools and Functionalities
CBR: Constant Bit Rate
CGI: Common Gateway Interface
CORBA: Common Object Request Broker Architecture
DCE: Distributed Computing Environment
ESIOP: Environment-Specific Inter-ORB Protocol
FTP: File Transfer Protocol
GDC: General DataComm
GIOP: General Inter-ORB Protocol

GPIB: General Purpose Interface Bus
HTML: HyperText Markup Language
HTTP: HyperText Transfer Protocol
ICMP: Internet Control Message Protocol
IDL: Interface Definition Language
IEEE: Institute of Electrical and Electronics Engineers
IIOP: Internet Inter-ORB Protocol
IP: Internet Protocol
IPX: Internet Packet Exchange
ISDN: Integrated Service Digital Network
JDK: Java Development Kit
JRE: Java Runtime Environment
JVM: Java Virtual Machine
Kbps: Kilobits per second
MB: Mega Bytes
Mbps: Megabits per second
NI-DAQ: National Instrument-Data Acquisition
OMG: Object Management Group
ORB: Object Request Broker
PSTN: Public Service Telephone Network
RDA: Remote Data Acquisition
RMI: Remote Method Invocation
RPC: Remote Procedure Call
SPX: Sequenced Packet Exchange
STF: Specific Tools and Functionalities
TCP: Transmission Control Protocol
URL: Universal Resource Locator
VDU: Visual Display Unit
VI: Virtual Instrument
VISA: Virtual Instrument Software Architecture
XML: eXtensible Markup Language

The section on “Interoperability of Heterogeneous Networks” summarizes current mechanisms used to insure interoperability among heterogeneous networks. The section on “Design of the Telecommunication Platform” presents the methodology adopted for designing the telecommunication platform proposed in this article. The section on “Implementing the Telecommunication Platform” discusses the implementation process, while the last section describes experimentation and analyzes results.

INTEROPERABILITY OF HETEROGENEOUS NETWORKS

To guarantee communications and information exchange between heterogeneous distributed networks, one has to have appropriate interoperability mechanisms. The following are among the most known interoperability mechanisms: S*ockets*, the HTTP/CGI set, S*ervlets*, RMI and CORBA (Bardout, Hauw, Pavon, & Thomas, 1998; Barton, Eykholt, Faulkner, Kleiman, Shivalingiah, Smith, Stein, Voll, Weeks, & Williams, 1992; Berners-Lee, Fielding, & Frystyk, 1996; Haggerty & Seetharaman, 1998; Henning, 1998).

Sockets

Sockets are communication points with names and addresses. They work as network access interfaces. Introduced in 1981 by way of UNIX BSD 4.2 version (Barton et al., 1992), they became exclusively part of the TCP/IP family of protocols. The TCP/IP family operates according to the following modes: connection oriented (TCP), connectionless (UDP), and natives (IP or ICMP). Later, special sockets extensions were added for other platforms and communication infrastructures such as IPX/SPX and ATM. This heterogeneity of contexts is also reflected in the variety of languages used for the programming of these communication points.

Even though an object oriented modeling of *sockets* facilitates the integration of client entities to servers in a distributed object environment, this type of modeling still has the following drawbacks:

- lack of transactions: there is no guarantee to insure the atomicity of operations involving distributed objects;;
- impossibility of distributed entities dynamic discovery, that is, an unavailability of means of updating and reaching connected and disconnected objects during a statically triggered session between two given entities; and
- non-existence of interface description, meaning that data taken from these links have no common representation, which means an increasingly low probability of successful remote operations involving computers of different types.

Thus, it becomes necessary to look for a higher level of programming with which this kind of deficiency can be masked.

HTTP/CGI

The HTTP/CGI brings to the Web, the possibility of invoking dynamic distributed operations using static HTML documents. Introduced in 1990, the HTTP protocol is in fact a call tool for remote procedures, a bit similar to the RCP on the top of TCP/IP (Berners-Lee et al., 1996; Bostica, Callegati, Cason, & Raffaelli, 1999). A typical client/server interaction through HTTP/CGI is represented in Figure 1.

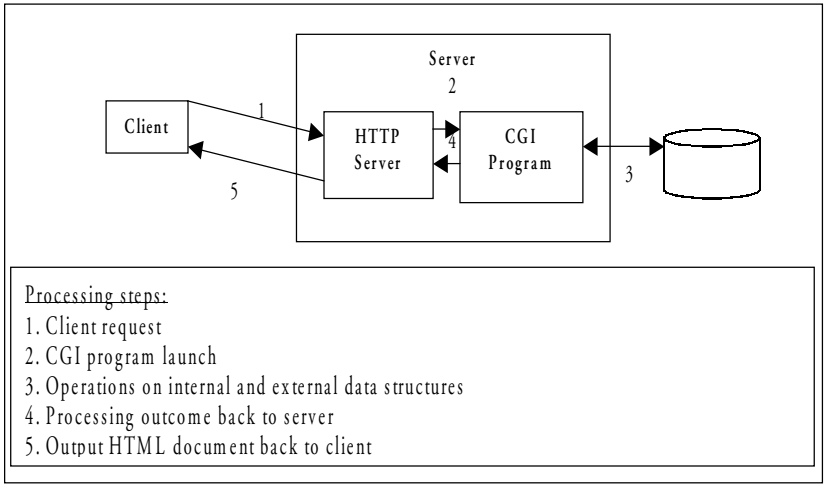


Figure 1. HTTP/CGI client/server interaction

In spite of its simplicity, the HTTP/CGI causes an enormous overload because successive requests from a single client are separately processed. The HTTP protocol is supplied with a web browser, which enables it to negotiate and manipulate data formats during a connection. The CGI protocol allows a Web server to launch any program with eventually input/output parameters. While, an object-oriented CGI code allows conveying distributed objects, such an option causes heavy overload because the HTTP mechanism is based on a highly interactive request/response scheme. The HTTP/CGI protocol contains two major flaws:

- Slow processing: could be considerable sometimes. It is caused by the multi-phase connection setup between a client and a server object;
- Complicated interactions: a sequence of requests/responses requires several connections between a client and a server.

Several negative aspects calling for the design of other solutions plague the HTTP/CGI protocol.

Servlets

In 1997, Javasoft introduced the first Java Web server supplied with a new type of tools called *servlet*. The server dynamically loads a *servlet*, which in turn supplies specific classes and modules capable on the one hand of modeling the CGI protocol, and on the other of coexisting with the HTTP protocol. In this case, the scenario is a bit more complex than previous cases, as shown in Figure 2.

Compared with the HTTP/CGI family of protocols, the *servlets* approach brings two improvements: a longer life cycle for a *servlet* and a possibility of a permanent connection to a database. This reduces the overload caused by successive requests, but increases the service duration.

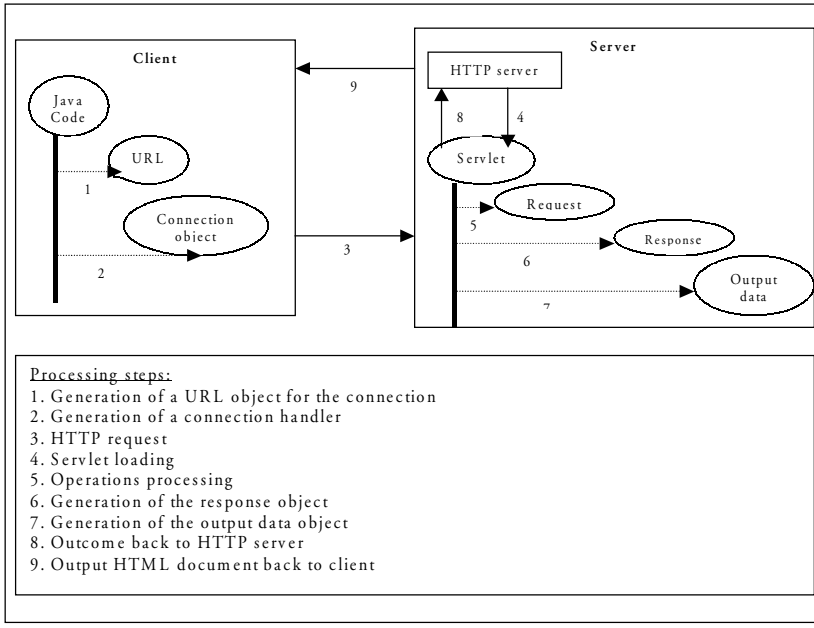


Figure 2. Client/server interaction using *servlets*

RMI

An API or RMI programming interface is nothing other than the integration of a model for distributed objects that can be manipulated locally or from a distance using Java virtual machines. Prior to any client/server interaction, with RMI, the following preliminary steps must be followed (Harkey & Orfali, 1998):

- the implementation of the server object;
  the generation of the real code for the object server (*skeleton*) and its delegated code or a client downloadable stub;
- the registration of the distributed object for public access; and
- the invocation of the distributed object.

To mitigate the system destabilization caused by downloading or subsequent exchanges, RMI requires the activation of a security manager on the machine as well as on the server. Typically, a client/server interaction under RMI takes the form described in Figure 3.

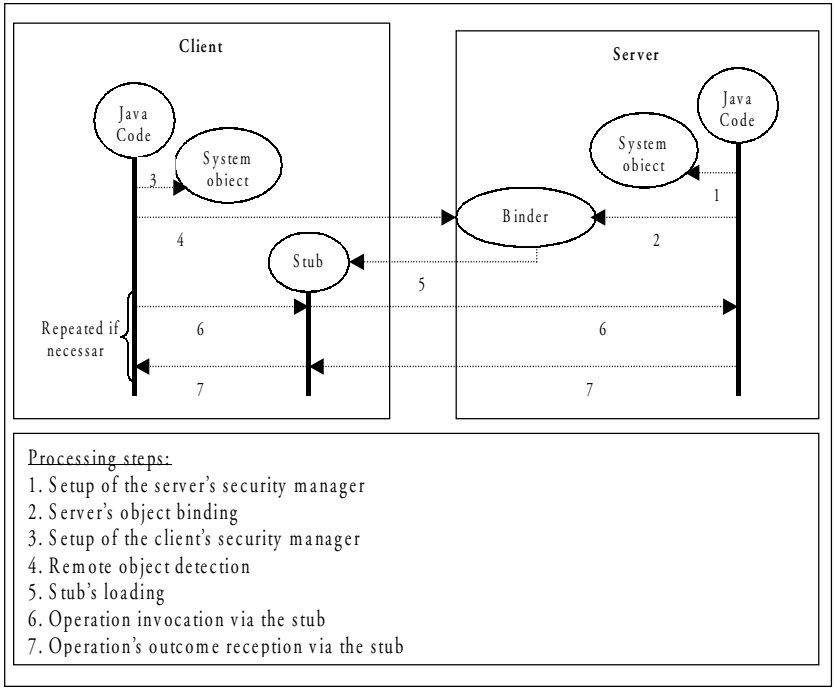


Figure 3. Client/server interaction using RMI

It clearly appears that the RMI approach differs from the others and brings the following advantages:

- quicker processing than with HTTP/CGI;
- an interface description for managing both client and server functions with a high level of abstraction;
- dynamic downloading secured by a code and Java classes; and
- a distributed object control including garbage collection mechanisms.

Moreover, RMI does not allow a propagation of transactions nor a dynamic discovery of distributed objects. RMI's major drawback remains its exclusively Java-based design, which eliminates the interoperability among objects generated with different computer languages. The possibility for the scalability of large distributed systems is thus being reduced.

In spite of these drawbacks, the RMI could be considered as a first step towards a more sophisticated concept for interactions in an open distributed objects environment. Javasoft and OMG have joined efforts to develop a more universal interoperability mechanism known as CORBA.

CORBA

CORBA is a family of specifications established by OMG (Seetharaman, 1998; Siegel, 1998; Vinoski, 1998). Two main aspects characterize CORBA's operational mode. On the one hand, every object must be provided with an IDL interface hiding the details of its real implementation, such as the operations performed by this interface, the input and output parameters of these operations, as well as the exceptions generated in the course of false executions. On the other hand, for both local and remote objects, static and dynamic requests are uniformly generated by routines and libraries forming ORB's layer at both ends of a connection (Schmidt, 1998). The IDL/ORB association keeps a high level of encapsulation, because there is no constraint on the programming languages used. Java, C, C++, Cobol, Smalltalk, and Ada already havee equivalent extensions in IDL. Figure 4 illustrates the typical interactions between clients and object servers.

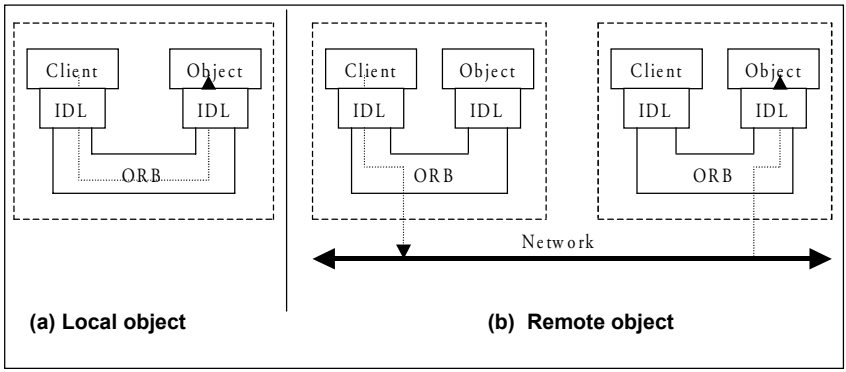


Figure 4. Client/server interaction using CORBA

One of CORBA's main features remains the fact that IDL object interfaces are independently accessible ~~of~~ regardless of used platforms and ORBs. Such mechanisms are made possible by a GIOP generic standard, which covers all interoperability aspects related to the physical data link and network layers. Designed to operate on the top of a transport protocol connection when it is used with TCP/IP, the GIOP standard leads to the IIOP protocol. However, nothing impedes the coexistence between GIOP and dependable protocols including asynchronous protocols such as IPX, ATM, and the SS7 family of protocols. In other contexts, protocols similar to GIOP, such as DCE/ESIOP (Harkey & Orfali, 1998), are defined.

CORBA applications are numerous and spread in multiple fields such as banking, commerce, education, health, and so forth. One of CORBA's classical applications is network management based on oriented object modeling of information management (Bardout et al., 1998; Haggerty & Seetharaman, 1998). Increasingly larger networks require more specialized distributed mechanisms. Hence, the use of CORBA becomes a necessity, since products such as GDC ProSphere management architecture for ATM networks are now available in the marketplace.

Java-CORBA Compatibility

Java is a simple object oriented language that is portable and dynamic. It has many elements common to both C and C++ languages. Java is, however, more sophisticated than these languages when one considers the security resources dedicated to execute its code. Java has the advantages of an interpreted language and guarantees the performance of a compiled language (Anuff, 1996). In effect, the compilation of a Java program provides for an intermediary code, known as *Java Bytecodes*, which could be interpreted on any platform. An interpreter is in fact a representation of what is known as *Java Virtual Machine* (JVM), whose implementation could be realized at the hardware level. Java development tools and web browsers are nothing but interpreters used for local applications and Java Applets, respectively.

With CORBA (Bardout et al., 1998; Haggerty & Seetharaman, 1998; Henning, 1998), Java is no longer an ordinary competitor to the HTML language for developing dynamic web pages. CORBA extends Java object models and increases their distributive aspects. Thus, while maintaining a light interface, Java Applets could use distributed components and trigger either dynamic operations or atomic transactions invoking several entities. Moreover, CORBA succeeded in introducing peer-to-peer communications among servers. As a result, we have on the one hand a Java programming style offering more services and, on the other, environments subject to higher scalability.

In conclusion, the CORBA/Java combination is more suitable for realizing certain critical aspects of telecommunication platforms. Like other distributed applications, virtual laboratories can benefit not only from efficient support for integrating specific and generic tools, but also interactive interfaces for flexible access. The design presented in the next section integrates these characteristics.

DESIGN OF THE TELECOMMUNICATION PLATFORM

This section presents the telecommunication platform model dedicated to support distributed virtual laboratories. First, the laboratory environment architecture is outlined, then, its functional and technical specifications are presented.

Architecture of the Laboratories' Environment

Generally, *telelearning* refers to the use of computer networks for learning (Ausserhofer, 1999; Collis, 1999; Collis, 1996). Telelearning allows for the supervision of students by trainers, tutors, and professors, all scattered in space and time. Thus, the support of learning activities in different scientific and technical disciplines requires distributed learning environments similar, as much as possible, to those in conventional laboratories.

The proposed inductive methodology consists of defining and developing different laboratories according to characteristics that are specific to the corresponding knowledge fields or disciplines. Then, the tools and functions on which diverse laboratories are based are regrouped into two large sets: BTF common to several types of laboratory, and STF that are rather peculiar to one type of laboratory. The generic feature of the virtual laboratory model results from the development of a sufficient number of specific laboratories integrating a variety of functionalities and tools. However, this generic aspect is mainly built on a telecommunication platform, that is, the physical support of BTF, as shown in Figure 5 (Kassouf Pierre, Levert, & Conen, 1999)

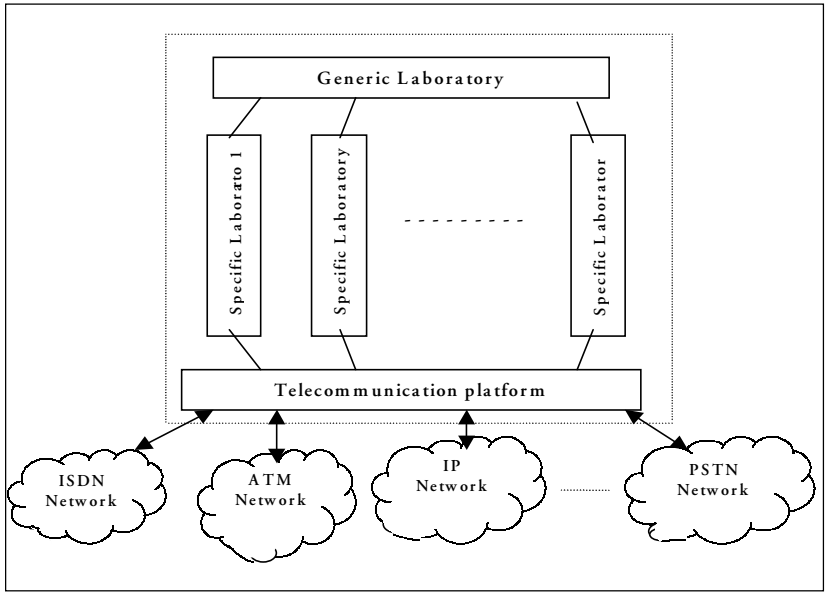


Figure 5. Proposed architecture for the virtual laboratories

Beyond the access to a variety of experimental manipulations, a telecommunication platform also guarantees interoperability among different access networks. Regardless of networks or telecommunication media, users could access a set of equipment supplied by each specific virtual laboratory. Depending on a user request, the platform sends back adequate specific tools with a copy of each commonly used tool. Thus, the user would be free to use functionalities without having to distinguish the generic from the specific.

The implementation of the proposed architecture is now being carried out through two laboratory prototypes: one for physics and the other for electrical engineering. The realization of other specific virtual laboratory prototypes is currently under way, in conformity with the proposed inductive approach.

Physics' laboratory. It supplies a multimedia environment including several simulations with sequences of video-recorded experiments, as well as animated and textual explanations. The implementation takes into account most significant aspects of a real laboratory beside practical experiment constraints such as measurement inaccuracy and lack of specification of fixed parameters. Each simulation could include its specific tools and other common tools borrowed from a generic basic model: notebooks, analysis, mathematical, and measurement instruments.

A physics virtual laboratory integrates access functionalities and displays multimedia, and remote manipulation of documents, as well as communication resources such as electronic mail and voice mail. Other more sophisticated functionalities could be added in the future. Online assistance during an experiment, shared tools, real-time control, and data exchanges could be enumerated as examples.

Electrical Engineering Laboratory. An electrical engineering laboratory has the same pedagogical assistance notions as those of a physics laboratory as well as similar types of interactions and manipulations. Moreover, it emphasizes basic electrical engineering concepts to which it adds a tele-experimentation functionality. Essentially, the objective is to carry out simulations of numerical models typically dedicated to the acquisition and processing of electrical signals. This kind of virtual environment offers the opportunity to use an equivalent of expensive laboratory equipment, whose real usage may sometimes put its manipulators at risk.

Beyond the distributed environment for applications which guarantees data sharing among participants, the electrical engineering laboratory also

supplies other types of tools including a videoconference. These tools supply among other things, new management and learning means for remote control and measurement that enable remote real-time operations. In this perspective, this virtual laboratory invokes modeling and simulation environments for the use of shared or non-shared hardware and software that one finds in traditional electrical engineering laboratories.

Telecommunication platform. In addition to providing access to virtual laboratories, the telecommunication platform manages to supply each specific laboratory with a package of sharable BTF~~ation~~ whose adaptation to any particular context remains completely transparent to the users. This model offers an example of conceptual and normative framework for~~to~~ exchanges among heterogeneous systems. It is also underpinned by the need for interoperability among several types of networks and communication media, particularly characterized not only from an architectural perspective, but also in terms of software and hardware tools. Therefore, it is the role of the telecommunication platform to mitigate the set of incompatibility problems that stems from ~~the~~ differences in protocols, data format, transmission flows, address formats, and so forth. The platform must also be capable of parallel processing. The latter is required for efficient sharing of instruments and remote measurements that are a frequently and strongly recommended feature for, and efficient usage of, sharable resources such as virtual instruments and measurement equipment. Besides, the following functions are added: control of coherence and integrity of transmitted data, flow and error controls usually performed over any communication network.

Generic model. It is the common origin of all specific laboratories and a necessary element to the definition of properties and attributes common to different types of laboratory. Each specific laboratory inherits a set of sharable tools and has specific tools. According to the proposed inductive method, the list of attributes is increasingly extended with the addition of specific laboratories. Identified attributes are classified into two groups: external attributes and internal attributes.

External attributes define features required for the working of laboratories and commonly used. Following is a sample of such attributes :

- support tools used for the preparation of experiment and during the experiments;
- measurement instruments such as rule, voltmeter, ~~l~~oscilloscope, and so on;
- remote experiment manager interfacing between client requests and answer servers;

- database link manager facilitating database access and consultation;
- communications tools allowing for efficient information exchanges such as electronic messaging, video-conference and audio-conference tools;
- reference tools used during experiments.

The second group of attributes includes attributes that are associated with the architecture and allow for presenting experiments. It is at the level of internal attributes that notions of experience, interface, and counselor system spaces appear. In this perspective, a key attribute emerges; it is called *experience manager*. That is the supervisor of the following five spaces presented to the experimenter:

- the *presentation* space which displays experiments in video and, other images;
- the *manipulation* space which allows for taking measures and manipulations;
- the *analysis* space which allows for analyzing results; and finally
- the *theoretical* and *application* space which displays a theory and its applications.

There is also the help manager which could be divided into two subsystems:

- the contextual help sub-system which supplies the user with help in a given situation such as the measure that should be taken with a given instrument; and
- the presentation sub-system which suggests the working method in a virtual laboratory.

Thus, a generic model allows for integrating in a coherent manner, an important number of BTF shared among several entities, possibly heterogeneous, through an evolutionary and adaptive platform.

Modeling the Platform

A telecommunication platform could be seen as a regrouping of modules. Each of these modules is dedicated to a specific task. As shown in Figure 6, these modules carry out the following three tasks: (a) adaptation to STF, (b) integration of BTF, and (c) adaptation to users' communication networks.

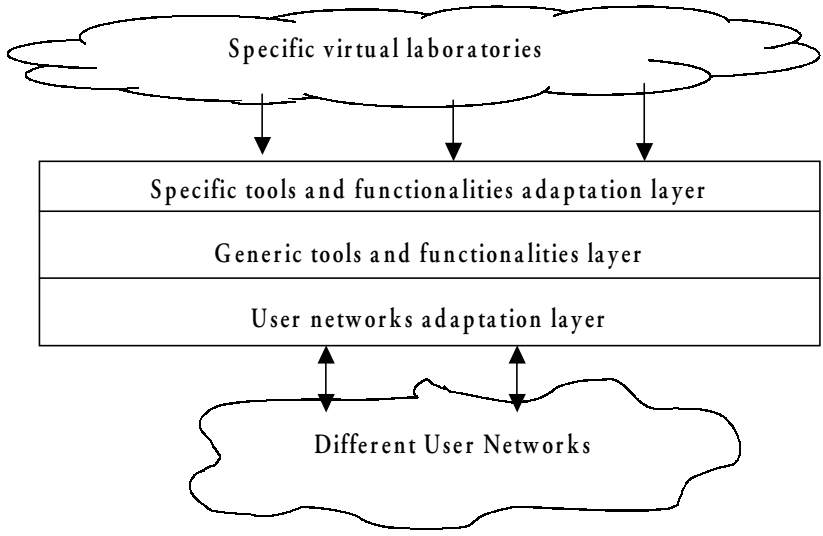


Figure 6. Triple-layered model for the telecommunication platform

This modular approach has the following advantages:

- *Independence*: the platform makes its resources available to virtual library applications without depending on a particular application;
- *Portability*: common tools could be added, withdrawn, or modified without altering the general output of specific laboratories;
- *Universality*: remote users are not constrained to a particular platform and not limited to proprietary computer systems; and
- *Modularity*: each of the platform's modules could be designed, implemented, and improved without the least change to the other modules.

Such an approach allows for an evolutionary development not only of the progressive design of BTF but also the adaptation of tools and functionalities that are specific to each type of laboratory. These features constitute the most original aspect of this platform.

To realize interoperability functions, the adaptation layer must meet the following requirements:

- *Operation in parallel mode*: is an ideal method to support a simultaneous access by several users. When the adaptation layer receives a

request, it launches threads capable of processing without interfering with other current processes.

- *Choice for an adequate communications*: For example, a client having an ATM access cannot be reached through a TCP/IP protocol. For each user, the network adaptation layer must invoke the interface and middleware encapsulating the communication protocol comprised and supported by its network.
- *User identification*: Virtual laboratory sites have public and private sections. The first section presents pedagogical information on the general objectives of virtual laboratories. The second section provides students with access to virtual manipulations. The distinction between the two categories of user requires the identification of each user, as well as his/her address and access code. This type of information is then stored on the platform, consulted and updated, independently of the communication means used by a client.
- *Quality of Service*: the adaptation layer realizes, among other things, the adaptation to the transmission speed of the physical support which links the network of each user to the platform. For an access by means of telephone lines or ADSL, the conventional transmission speed is of a few hundreds Kbps. But, for an ATM network, whatever the class of the negotiated traffic by the client during the setting of the link CBR or ABR, the transmission could reach a million of bits per second (Mbps). At the level of the adaptation layer, the envisioned method for this type of situation consists of compression and decompression mechanisms applied to data before and after they pass through the user network's adequate stack of standard communication protocols. The algorithms as well as the rates of compression vary according to the nature of each network.

When a client asks for a particular manipulation and selects the adequate parameters, the adaptation layer assigns to this request, a session where all previous exchanges regarding the client's experimentation could take place in a simple and transparent way. The user will be concentrated on his/her experiment without noticing the events that occur on the platform or on his/her own machine. Besides interoperability, the adaptation layer takes into account functions in any network environment such as the management of clients' requests, security, administration of virtual laboratories' resources, and so forth.

The BTF layer provides a list of basic tools commonly integrated into different specific virtual laboratories. What is shared among specific laboratories could be assimilated to structural groupings, each having features that

distinguish it from the others. Each structure is considered, from an object-oriented standpoint, as a class whose instances have their own attributes and exclusive manipulation methods. An example is the measurement function translated into a group of classes, each corresponding to a measurement instrument such as a barometer, a protractor, or a voltmeter. Thus, the tools are nothing but objects derived from these classes. However, each tool also has specific attributes and private methods that are internal procedures, thereby invisible to users. Sometimes, basic tools require means to contact other objects scattered in the same experimental framework. This type of interaction could be realized by means of public methods.

Adaptation to STF is a necessary condition for an efficient experimentation focusing on users. While maintaining a class and object perception to identify function and tools, the adaptation layer does not present new classes and structures. Rather, it allows for the cooperation between STF and BTF in a given specific laboratory. An example of this is the tuning of the weighting scale that is necessary for a physics laboratory where masses vary from a few to hundreds of kilograms. As for a chemistry laboratory, manipulated matters often vary from milligrams to dozens of grams. Moreover, an example for electrical engineering is the time scale of a chronometer which is tuned at microseconds or milliseconds for controlling signals. This chronometer is regularly and automatically triggered and it starts anew, any time it registers the duration of an action by an enzyme or by a catalyst in a biological metabolism.

The adaptation to STF essentially consists of manipulating the public parameters of common objects. An adjustment of certain variables, a return to zero of others, a new grading of measurement instruments and a new activation of more advanced graphic options are a few samples of requested tasks at the level of the telecommunication platform's first layer.

IMPLEMENTING THE TELECOMMUNICATION PLATFORM

A specific laboratory could be considered as a collection of distributed objects. Interactions among these objects and between these objects and external basic tools follow well-defined rules and protocols according to the considered scientific discipline. At the moment, the platform only supports two laboratory prototypes, one for physics and the other for electrical engineering.

Prototypes Implementation Details

The physics laboratory is a reproduction of a real laboratory provided with many measurement instruments and experimental materials. It was developed with a *Micromedia* product, known as *Director*. The used programming language *Lingo*; is an object-oriented language which allows for powerful interactions based on combinations of graphics, text, sound and video.

In the case of an execution in local mode, the virtual laboratory is a file executable with the extension (.exe); its distribution on a communication network is realized in two ways. The first way is considered to be primitive and consists of downloading the executable code that is launched on a client site, eventually after having carried out a few security configurations. The second way is more sophisticated and calls upon another *Macromedia* product, called *Schockwave.* That is an intermediary module authorizing, a web browser for triggering an execution of a program generated with *Director* in the form of web pages. More specifically, it is an Internet engine dedicated to the distribution of multimedia applications on a weak bandwidth. Obviously, *Schockwave*'s success is due to the fact that *Director* is supported by a set of Internet standards, an integrated set of support protocols such as HTTP and FTP, as well as other functions including the XML document interpretation and the HTML file integration. In this context, bidirectional data exchanges are also possible, using the support interface CGI. An important function consists of automatically converting the *Lingo* code into a Java Applet, easily understandable by any JVM. This is an Internet distribution mechanism equivalent to *Shockwave*.

However, the two suggested methods have some drawbacks. While the downloading of the code to be run could be very slow, for the local filing system to be explicitly accessible, the user must establish special permissions. As a result, the portability and flexibility of a remote execution are reduced, especially when the platforms supplied with UNIX and LINUX operating systems are excluded. This drawback is also the case with *Shockwave*, designed especially for Windows 95/98/NT and Apple OS 8 operating systems. The conversion in Java allows for partially overcoming this problem, because an interpreter of a Sun station captures the resulting Java code. However, this alternative is also considerably slow and incompatible with a few of *Lingo* instructions. Another means was considered for establishing a link between a physics laboratory's BTF written in Java. This is feasible through the use of the intermediary JavaScript code for reshaping exchanged data flows with *Schockwave*. However, realizing this choice appeared to be complex and not dependable.

The electrical engineering laboratory displays conventional aspects of electrical engineering, particularly regarding the processing of electrical signals. The shaping of virtual electrical engineering prototypes could easily be undertaken by using *Labview* system. This system is built by *National Instrument*. It is a parallel, graphic, and multitask programming system. The objective is to regroup in *virtual instruments*; varied electronic components such as gauges, thermometers, lamps, interrupters, and so forth. The interface built with the graphic language G is compiled to provide an executable code to a comparable speed with that of a compiled code C. Virtual or real electrical instruments involved in an application are managed by means of manipulated standards controllers through *Labview* libraries and standard interfaces such as VISA and GPIB.

Connectivity to remote instruments could be seen from different angles. For an electrical engineering laboratory, three alternatives are predominant. The first alternative directly corresponds to data capture and physical access to distributed electronic equipment. The result is an executable file that allows for capturing and analyzing received or transmitted signals through a GPIB interface according to the IEEE 488 standard. GPIB supports diverse communication mechanisms among which the TCP/IP and ATM architectures. This approach does not impose a limit on the number of simultaneously connected users.

The second alternative consists of a client/server model dedicated to data capture in input/output at the level of instruments on a local or extended network. This necessitates packages of NI-DAQ software from the client's side as well as from the server's side. The process of remote access used is called RDA and uses the TCP/IP family of protocols. In other respects, a scenario for distance data capture is completely transparent with RDA and could be implemented after an adequate configuration of the client entities in order for them to recognize the invoked server. Thus, a client will be capable of receiving a signal issued from one or several servers at the same time. However, an important obstacle remains on the server side because its capacity to serve several competing clients is reduced and runs the risk of reaching only one client at a time.

Finally, *Labview* has a development tool for Internet, known as *Internet Development Toolkit for G*. This tool allows for building HTTP servers capable of converting virtual instruments into recoverable images in HTML documents. These images could be seen through web browsers. Special mechanisms could be used to insure the security of these HTML pages. Moreover, this programming tool offers the possibility of generating virtual instruments encapsulating the CGI code for more dynamic exchanges of

requests between clients and servers. Virtual instruments are also supplied with more classic instruments such as electronic mail and file transfer to an FTP server.

Access to virtual laboratories is in fact an access to a website hosted in an HTTP server. The welcome pages offer the user an overview of the laboratories' pedagogical objectives. The surfing is undertaken by means of HTML links. Some of these links also serve as starting keys for experimentation. These pages launch laboratory on a client's platform as well as on an autonomous VDU for basic tools.

In order to launch a laboratory, the codes are activated according to the designated prototype. For the physics laboratory, the *Shockwave* tool has been used to export the *Lingo* code in a HTML format. In return, the electrical engineering laboratory uses the remote RDA access procedure.

The VDU that presents the basic tools is a Java Applet which integrates among other things, a chronometer, a notebook, and a calculator. The adaptation of these tools to the specific context of a laboratory is done during the downloading of the Applet. In other words, every specific laboratory has its own call interface comprising this type of Java Applet. Figure 7 illustrates the implementation of the BTF layer for the design model, as well as the adaptation layer to STF.

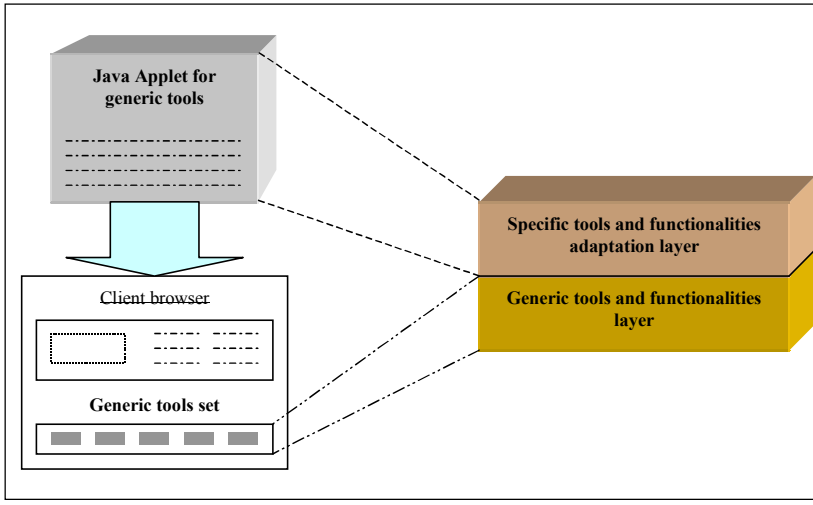


Figure 7. Implementation of the first two layers of the conceptual model

The layer for the adaptation to users' networks is implemented with a Java Applet that calls upon the HTTP interface and the TCP/IP family of protocols. Besides the HTTP interface, a generic tool that specifies the exchange of requests according to the CORBA architecture is not impeded by the heterogeneity of communication networks. The underlying architecture often lays on the IIOP protocol. It allows the ORB to efficiently carry out remote operations and communicate the results to clients. Interoperability is guaranteed in a transparent way to the user, as shown in Figure 8.

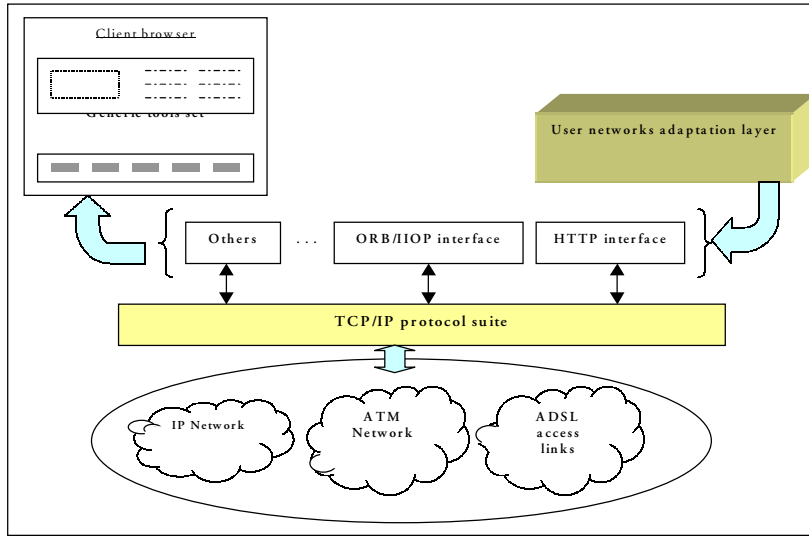


Figure 8. Implementation of the user networks adaptation layer

The Java Applet consists of a process which creates a child process specific to each invoked tool. As long as the tool in question is active, the loading process remains active in the back. The logical structure of the Java Applet is represented in Figure 9. The chronometer (a set of Java classes) records temporal delays through an internal clock of the client site, without an intervention from the server. The scenario is identical for the calculator that calls upon more advanced graphic functions of Java. The notebook is a representation of a database devoted to virtual laboratories. It is a table on which clients access to experimentation sessions. A typical access to the notebook is illustrated in Figure 10.

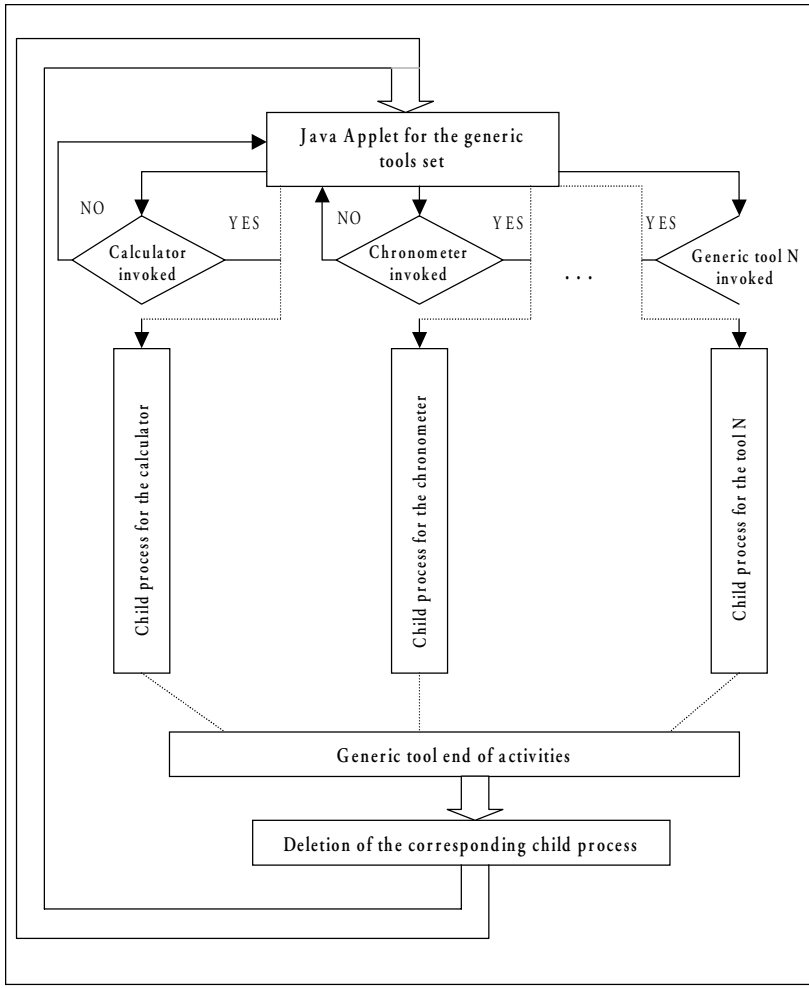


Figure 9. Processing scheme

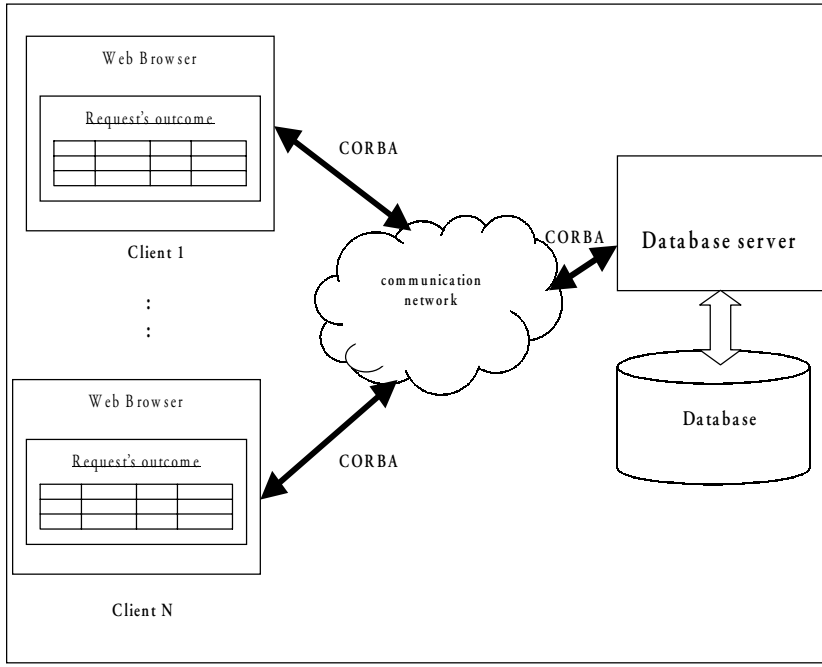


Figure 10. Notebook access scheme

Environment and Implementation Tools

The basic tool VDU of the platform is itself a Java Applet inserted in a HTML page, downloaded from a public HTTP server. It is capable of functioning with any operating system if, and only if, the operating system supports a web browser. However, at the level of the client system, the actual structures of laboratory prototypes allow for the following specifications:

- Windows 95/98/ NT 4.0 or more as well as MacOS (Power PC) as operating system;
- Pentium II (or better) as processor;
- A core memory of 32 MB minimum (64 MB or more is recommended);
- A disk space of 125 MB minimum; and
- Netscape Navigator 3.0, Internet Explorer 3.0, or their latest versions are recommended browsers.

The client must have on the machine the complete version of the execution environment JRE 1.2 (Java 2 platform). In effect, JDK 1.2 is a development tool which, for the moment, is not part of current versions of browsers. These browsers are provided with versions prior to JDK 1.1. In relation to JDK 1.1, significant improvement are brought to JDK 1.2: a better graphic interface, more secure access to a better control policy, more advanced RMI mechanisms, a CORBA support for interoperability, and connectivity to remote services.

To get the JDK 1.2 written Applets started within different existing web browsers, a Java plug-in is advised. It allows the web browser execution environment for abstaining, while

it provides the Java code with a Java virtual machine (JVM) capable of interpreting instructions previously generated by a Java JDK 1.2 compiler. When the JRE 1.2 is installed on the client site, in order to activate the JRE environment, it is sufficient to invoke the Java extension module during the downloading of the Applet by way of the HTTP standard interface.

As for the CORBA support, a Java ORB compatible with CORBA/ IIOP 2.0 specifications is supplied in the form of JRE libraries. Through the Applet, certain generic tools could initialize an ORB and later connect to every distributed object, identifiable by a unique object reference (IOR).

To establish a CORBA connection and manipulate the remote object, a configuration of the security policy is necessary for the locally downloaded Applet. This step is carried out once only, using the JRE 1.2 *policytool*. However, the specification required from the client is not complete and every time is subject to addition of new extensions according to products used for virtual laboratory prototypes. In this category, one can find *Shockwave* for the physics laboratory or other *Labview* special extensions for capturing and controlling data manipulated by distributed virtual instruments.

As for the server, the problem is a bit delicate because of the multiplicity of generic and specific entities serving clients. In effect, the inductive approach adopted for the design of the generic virtual laboratory model is such that the addition of a new laboratory introduces at least one additional computer tool. As shown in Figure 11, several servers could be involved in this context. Such is the case, for example, of the electrical engineering laboratory where the client controls an engine from a distance in a direct link with a *Labview* server and a determined protocol.

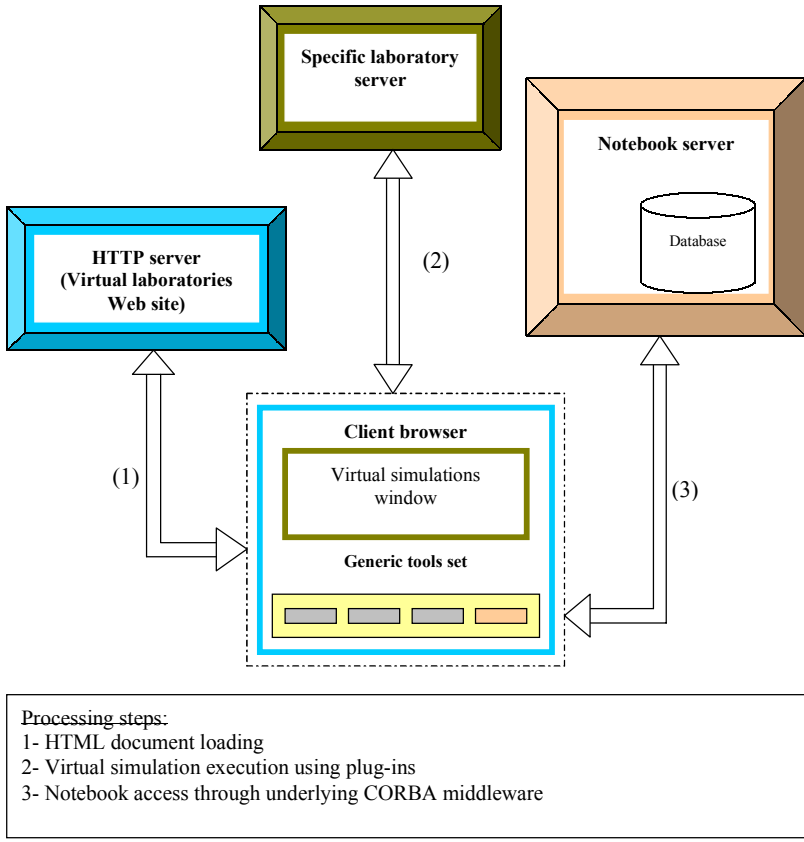


Figure 11. Typical virtual laboratory manipulation

Some generic tools that are part of the distributed CORBA object environment also require dedicated servers. In this case, the objects themselves act as servers to which any client access is carried out by respecting CORBA's architecture operating model. In this realization, the notebook supplied by the telecommunication platform refers to this type of server.

HTTP servers form the third category of servers. They are also called web servers. An HTTP server distributes HTML documents corresponding to virtual laboratories. The setting of the web site dedicated to this project includes an opening page, information on the technical and pedagogical aspects of specific laboratories, adequate means for launching virtual simulations, HTML links referring to other sites, as well as servers that could coexist

on the same or different machines. Their positioning is a function of hardware and software features. For any HTTP server, the required tools are HTTP tools that could exist on platforms and environments of different natures, such as UNIX and Windows NT workstations.

EXPERIMENTATION RESULTS

The performance measurement of the telecommunication platform could be based on criteria such as: dependability of links, client machine's operating features, and servers' capacity. The flexibility of a web browser allows for hiding the complexity related to the adaptation and integration of the entire set of tools used.

Browsing Mechanisms and Interface

The homepage of the virtual laboratories' site is shown in Figure 12. This site includes public and private sections distributed on different laboratories. The adaptation layer dedicated to users discriminates between authorized and nonauthorized users. It is connected to a central controller linked to a database. When access is authorized, the user can start manipulating. A new page appears on the user's VDU, showing a list of available manipulations and generic tools.

The VDU of generic tools is present during every manipulation. It supplies access to basic tools by means of buttons. It is necessary to push on one of these buttons to activate the corresponding tool. Options currently implemented include the chronometer, the calculator, and the notebook.

The chronometer. Once this button is selected, the temporization is triggered. An "Active Chronometer" then replaces the button labeled "chronometer." It is sufficient to push on this button to stop the operation and return to the initial state. The registered temporization is displayed on the contextual field.

The calculator. Pushing this button activates a scientific calculator. Once the corresponding window is opened, the user will have the choice to carry out some elementary algebraic or trigonometric operations. Other options such as memorization of results, change of angular units, and the trace of polynomial functions of degree equal to or lower than 5 are also possible. The current version of the calculator is shown in Figure 13.

Figure 12. Virtual laboratories' homepage

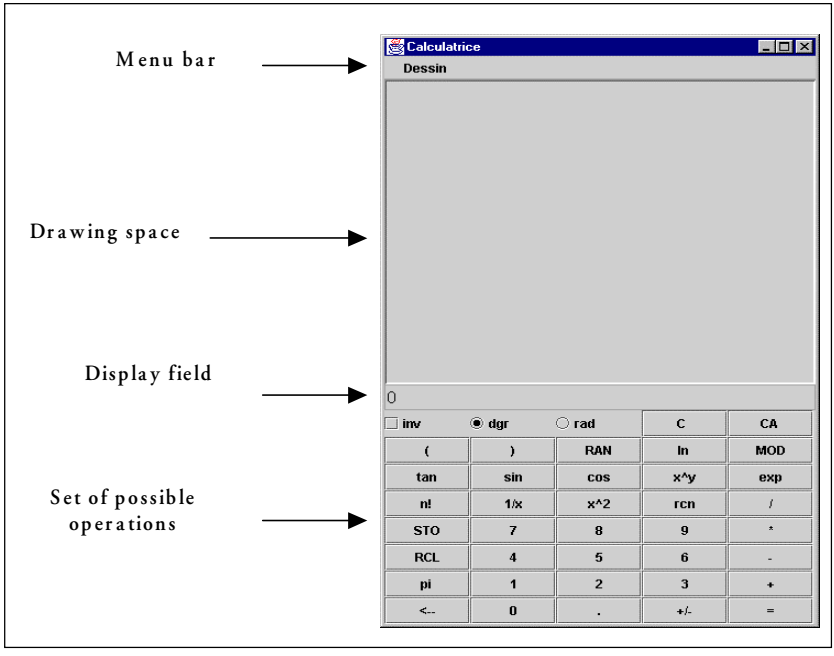


Figure 13. Generic calculator

The notebook. The flexibility of processing information stored in databases offers an ideal choice for memorizing the output measures taken within diverse categories of simulation. Each manipulation is identified by a unique code. As well, each student attending a class is also identified by a unique code. These two codes allow for finding a student's experiment results during a particular laboratory session. At the beginning of every access to a notebook, the student code and the manipulation code are captured. As early as the student and the manipulation carried out are known, corresponding results are displayed on the screen, as shown in Figure 14.

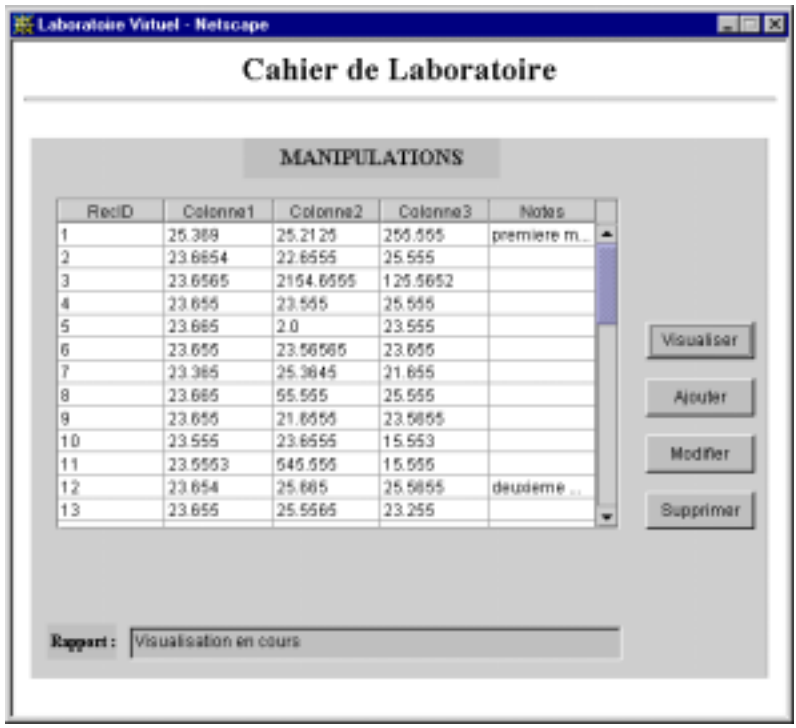


Figure 14. Notebook contents

Experimentation of the Telecommunication Platform

In order to evaluate the platform's performance, we have proposed a few experimental scenarios based on a certain communication infrastructure. Workstations are distributed throughout three geographical sites, distant

from one to another and connected to each other by communication links of different types. Sites 2 and 3 are linked in a local mode forming with Site 1 an extended (metropolitan area) network. Figure 15 shows the site's structure and the nature of communication supports that link these sites.

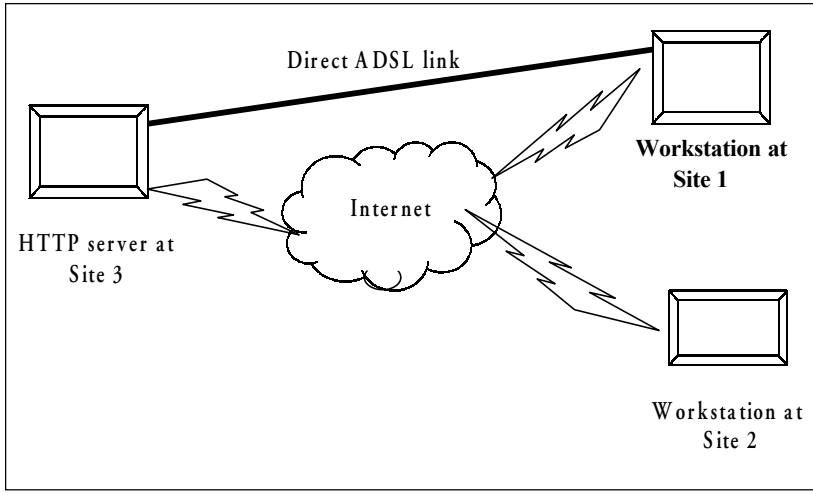


Figure 15. Experimental infrastructure

The performance of this telecommunication platform has been evaluated by comparing graphics illustrating delay variation and number of exchanged TCP/IP packets during the HTTP downloading of the entire generic and specific tools of the physics laboratory, as well as during the manipulation of a notebook by way of CORBA. Table 1 presents the adopted notations for the test sessions.

Table 1
Test Session Notation

Day Period	Site 1 (Client)	Site 3 (Server)
Morning	**P1**	**L1**
Afternoon	**P2**	**L2**
Evening	**P3**	**L3**

Figures 16 and 17 present respectively, the delays and the number of corresponding packets, by using the Internet to access the physics laboratory which integrates basic tools, textual information, and video sequences of manipulations.

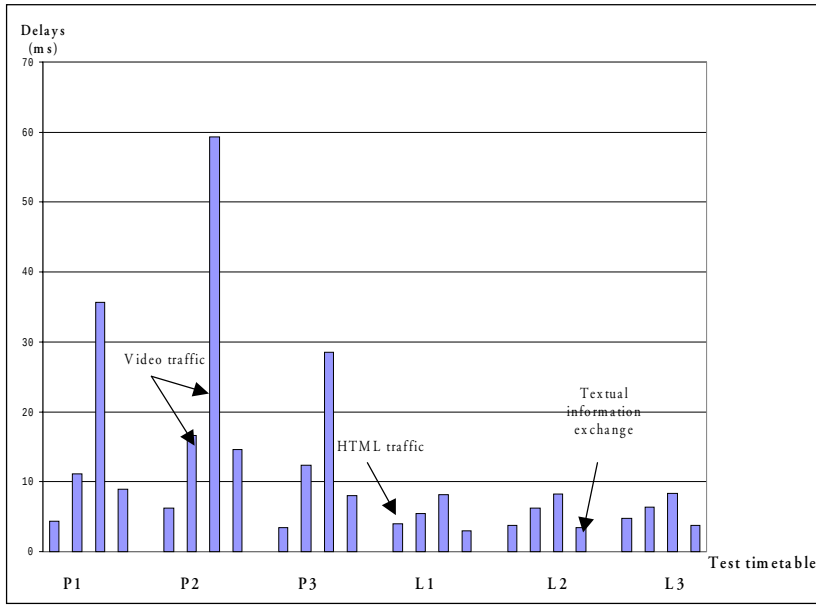


Figure 16. Physics laboratory loading delays

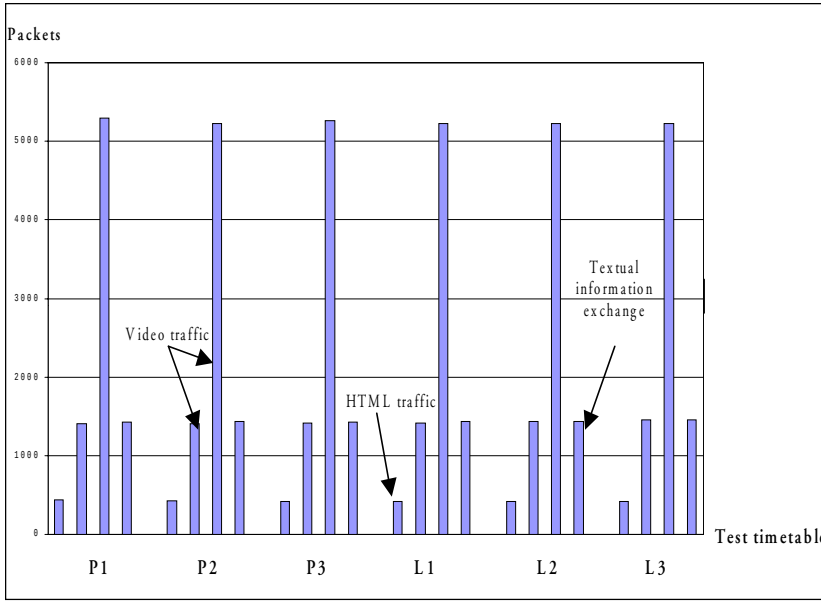


Figure 17. Physics laboratory overload

The physics laboratory. Whatever the period and experimental site, the physics laboratory requires approximately the same number of TCP/IP packets whose size varies between 60 and 300 bytes. As shown in Figure 17, the latter fluctuates around 450 packets for HTML pages, 1400 packets for a first video sequence, and 1400 other packets for the manipulation of a table with an air squab having mainly contextual information.

In return, the transmission delay between a client and the web server installed on the Site 3 depends on two key factors: local or extended client network, and the load of this network. For the first factor, the series of measures L1, L2, and L3 of Figure 16 present delays practically identical to a given type of media. The access period to the physics laboratory does not matter. In this case, the client and the HTTP server are both weakly loaded in a local network. The network bottleneck appears in the series of measures P1, P2, P3 where the client is the Site 1. For the four types of media, the delays are higher than previous values by a ratio of 2 or even 3. The longest delays are observed, particularly in the afternoon when the links are most loaded.

Regarding CORBA exchanges, we have erected similar graphics to previous ones. These graphics are distributed by the notebook's operation type. The chronometer and calculator are implicitly loaded with the physics laboratory. The use of these is perfectly static and does not call for dynamic exchanges. However, the manipulation of the notebook requires dynamic communications with the database server, which is, placed apart on a machine. Figures 18 to 22 respectively show the cost on Internet of some operations such as the opening of a notebook, the addition of an record to a database, the modification and suppression of an existing record, as well as the closure of a notebook.

The notebook. TCP/IP packets encapsulating information bytes as well as control bytes relative to the IIOP/CORBA protocol circulate during dynamic exchanges between a client and a database server. Each command to the database introduces different delays and number of packets. For the series of measure L1, L2, and L3, the number of packets and the delays are slightly higher than those of the series P1, P2, and P3 realized in a local network. However, the limited number of bytes of each request minimizes the influence of the network's load in both cases. In an extended network, losses of packets and retransmissions could occur, which explains the growth in the number of packets and delays. Test results are represented respectively in a local mode in Table 2 and in an extended mode in Table 3.

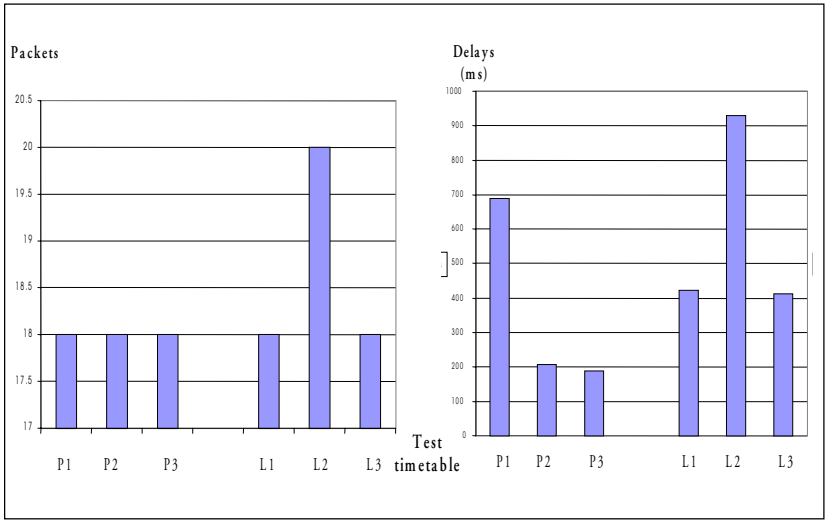


Figure 18. Connection establishment and overload delay

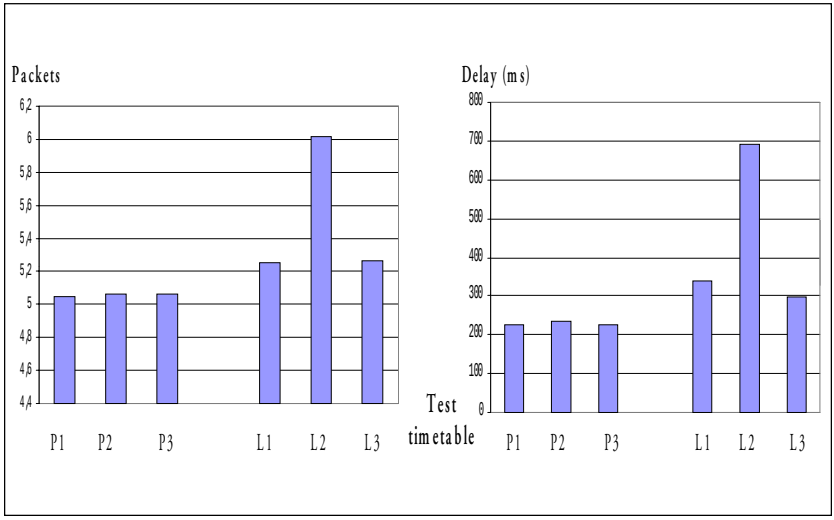


Figure 19. Record insertion overload and delay

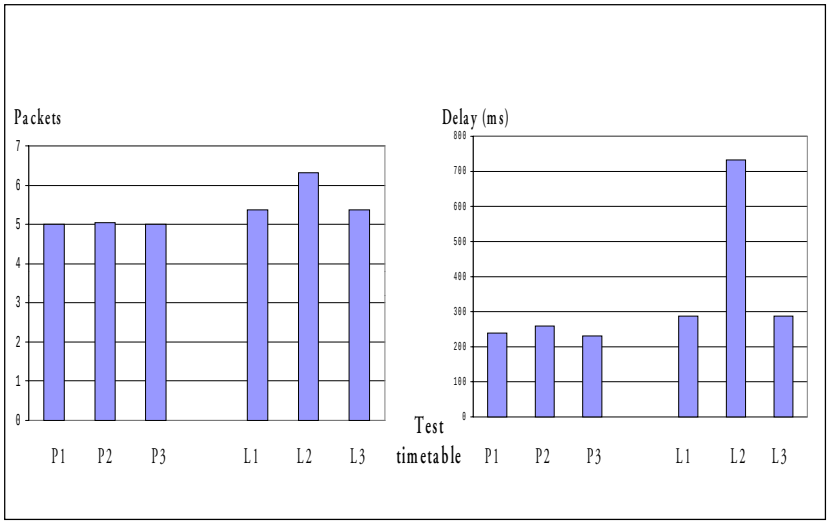


Figure 20. Record updating overload and delay

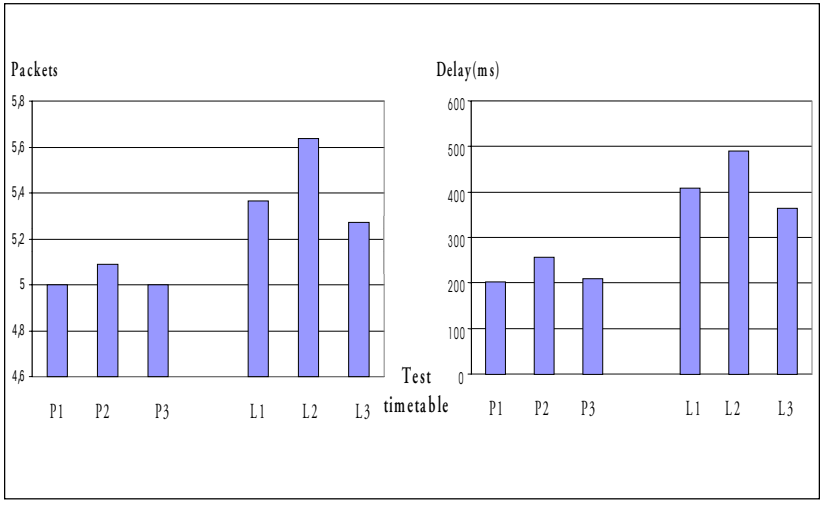


Figure 21. Record deletion overload and delay

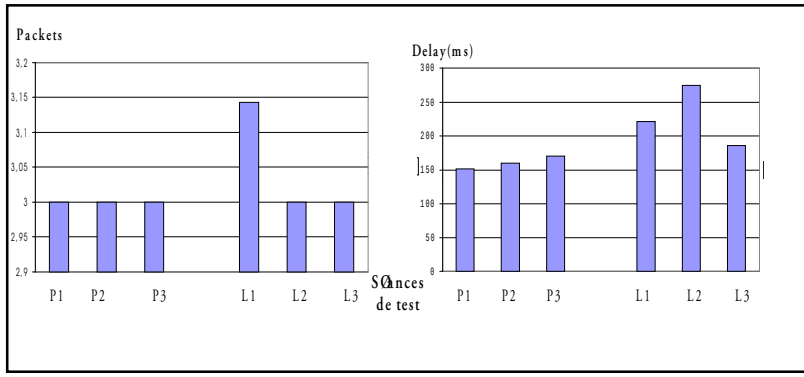


Figure 22. Disconnection overload and delay

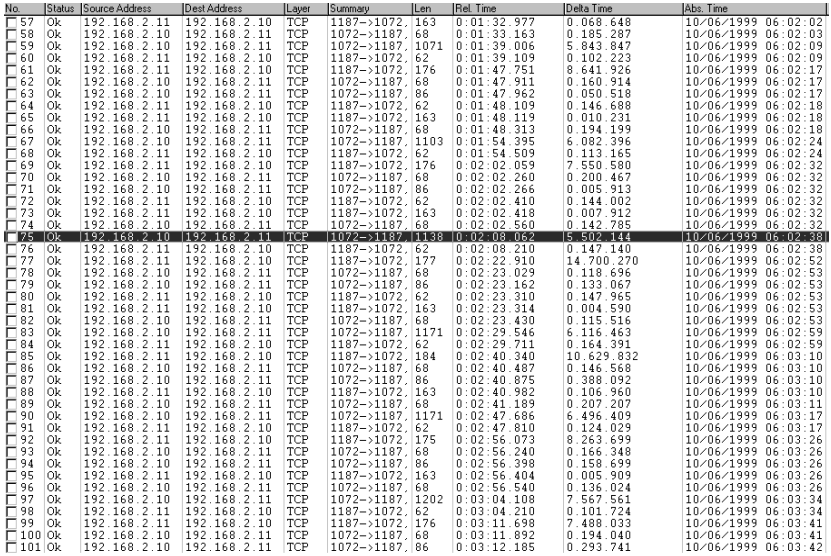
Table 2
Manipulation Results of the Notebook in Internet Mode

Average Value	Number of Packets	Delay (ms)
Opening of the notebook	18	360
Addition of a recording	5	220
Update of recording	5	230
Deletion of a Recording	5	220
Closure of the notebook	3	160

Table 3
Notebook's Manipulation Results in an Extended Mode with Internet

Average Value	Number of Packets	Delay (ms)
Opening of the laboratory Book	19	560
Addition of a recording	5.5	430
Updating of a recording	5.6	430
Deletion of a recording	5.4	420
Closure of the notebook	3.04	220

The telecommunication platform on the ADSL's direct link joining the client's station of Site 1 to the HTTP server of Site 3 which is installed, in this case, on the same machine as the database server was also tested. Table 4 presents a sample of statistics realized on exchanged requests. In this particular case, the period of test does not matter, because we use a closed circuit on which no other user can be hooked. Figure 23 results are summarized on Tables 4 and 5.



No.	Status	Source Address	Dest Address	Layer	Summary	Len	Rel. Time	Delta Time	Abs. Time
57	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 163		0:01:32.977	0.068.648	10/06/1999 06:02:02
58	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:01:33.163	0.185.287	10/06/1999 06:02:03
59	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 1071		0:01:39.006	5.843.847	10/06/1999 06:02:09
60	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:01:39.109	0.102.223	10/06/1999 06:02:09
61	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 176		0:01:47.751	8.641.926	10/06/1999 06:02:17
62	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:01:47.911	0.160.914	10/06/1999 06:02:17
63	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 86		0:01:47.962	0.050.518	10/06/1999 06:02:17
64	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:01:48.109	0.146.688	10/06/1999 06:02:18
65	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 163		0:01:48.119	0.010.231	10/06/1999 06:02:18
66	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:01:48.313	0.194.199	10/06/1999 06:02:18
67	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 1103		0:01:54.395	6.082.396	10/06/1999 06:02:24
68	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:01:54.509	0.113.165	10/06/1999 06:02:24
69	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 176		0:02:02.059	7.550.580	10/06/1999 06:02:32
70	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:02.260	0.200.467	10/06/1999 06:02:32
71	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 86		0:02:02.266	0.005.913	10/06/1999 06:02:32
72	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:02:02.410	0.144.002	10/06/1999 06:02:32
73	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 163		0:02:02.418	0.007.912	10/06/1999 06:02:32
74	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:02.560	0.142.785	10/06/1999 06:02:32
75	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 1148		0:02:08.062	5.502.145	10/06/1999 06:02:38
76	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:02:08.210	0.147.140	10/06/1999 06:02:38
77	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 177		0:02:22.910	14.700.270	10/06/1999 06:02:52
78	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:23.029	0.118.696	10/06/1999 06:02:53
79	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 86		0:02:23.162	0.133.067	10/06/1999 06:02:53
80	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:02:23.310	0.147.965	10/06/1999 06:02:53
81	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 163		0:02:23.314	0.004.590	10/06/1999 06:02:53
82	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:23.430	0.115.516	10/06/1999 06:02:53
83	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 1171		0:02:29.546	6.116.463	10/06/1999 06:02:59
84	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:02:29.711	0.164.391	10/06/1999 06:02:59
85	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 184		0:02:40.340	10.629.832	10/06/1999 06:03:10
86	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:40.487	0.146.568	10/06/1999 06:03:10
87	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 86		0:02:40.875	0.388.092	10/06/1999 06:03:10
88	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 163		0:02:40.982	0.106.960	10/06/1999 06:03:10
89	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:41.189	0.207.207	10/06/1999 06:03:11
90	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 1171		0:02:47.686	6.496.409	10/06/1999 06:03:17
91	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:02:47.810	0.124.029	10/06/1999 06:03:17
92	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 175		0:02:56.073	8.263.699	10/06/1999 06:03:26
93	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:56.240	0.166.348	10/06/1999 06:03:26
94	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 86		0:02:56.398	0.158.699	10/06/1999 06:03:26
95	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 163		0:02:56.404	0.005.909	10/06/1999 06:03:26
96	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:02:56.540	0.136.024	10/06/1999 06:03:26
97	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 1202		0:03:04.108	7.567.561	10/06/1999 06:03:34
98	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 62		0:03:04.210	0.101.724	10/06/1999 06:03:34
99	Ok	192.168.2.11	192.168.2.10	TCP	1187->1072, 176		0:03:11.698	7.488.033	10/06/1999 06:03:41
100	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 68		0:03:11.892	0.194.040	10/06/1999 06:03:41
101	Ok	192.168.2.10	192.168.2.11	TCP	1072->1187, 86		0:03:12.185	0.293.741	10/06/1999 06:03:42

Figure 23. Exchanged requests between a client and a server on an ADSL link

Table 4

Downloading Results for the Physics Lab with ADSL

	HTML pages	Video 1	Video 2	Textual Information
Number of Packets	445	1408	5459	1545
Number of Bytes	424728	1411097	5595294	1561197
Delay (s)	10.068	21.937	65.533	39.992

Table 5

Manipulation Results for the Notebook with ADSL

Average Value	Number of Packets	Delay (s)
Opening of the notebook	37	29.119
Addition of a recording	8.5	10.761
Update of a recording	8.5	11.278
Deletion of recording	8	8.751
Closure of the notebook	3	0.164

The performance of the ADSL link is in fact far from our expectations. It is comparable with that of a heavily loaded Internet network. Although our link works with a speed that is between 600 and 880 kilobits per second (Kbps), the main cause for the increase in delay in relation to the Internet case is due to the transmission of a packet whose size is often over one 1000 bytes from the server. This packet accompanies each request, except the interruption request. This type of packet could be seen on the shaded line of Table 4. It corresponds to the growth in the number of packets exchanged during the manipulation of the notebook. As for the different flows of media, the numbers of transmitted packets vary reasonably in relation to the previous case. However, the number of exchanged bytes is bigger and sometimes could triple the previous case. This confirms the presence of packets on the ADSL link. This packet is very large and sometimes could be of 1000 bytes, as shown in Figure 23.

CONCLUSION

In this article, a telecommunication platform for supporting distributed virtual laboratories has been presented. Three layers compose the architecture of this platform. The first layer takes into account the notions of access to the laboratories and interoperability among heterogeneous networks allowing users for accessing virtual laboratory environments. The second layer supplies a set of tools and generic functions sharable among several specific laboratories. The third layer insures the adaptation of these basic tools to other specific tools that exist in the peculiar context of each laboratory.

The authors' concern was to design an open platform, in the sense that different categories of technologies can coexist and evolve. The result is a distributed system in which diverse elements inter-operate while invoking varied computerized environments and tools. Thus, the access to the website for virtual laboratories includes not only pedagogical contents but also access mechanisms to simulations and virtual experimentation from different specific laboratories. During the whole laboratory session, the student could use activated generic tools from a VDU giving access to the laboratory. The actual configuration involves several servers distributed throughout distinct geographical sites: a HTTP server, a server for managing notebooks, and several other servers to manage specific laboratories. Some preliminary configurations are required on the client site to enable the use of all implemented computer modules. Different access links allow for connecting a client to all servers on the platform.

To validate the implementation, a series of interactive tests at different hours during a day, involving different access links was carried out. Experimentation scenarios have been elaborated in order to evaluate the performance of communications during the use of generic tools and specific laboratories. For this, the static transfer aspect of HTML documents of Java classes, as well as the dynamic exchange that normally takes place during the use of a notebook was considered.

The analysis of results reveals the existence of an enormous traffic that is generated during the downloading of HTML pages, particularly video sequences. Then, dynamic exchanges allow only a determined number of TCP/IP packets. The transmission speed differs from one type of link to another. With the Internet, this is translated into variable delays which remain smaller than those observed on the specialized ADSL link. The network congestion and the quality of service required by the user are factors determining the global efficiency of this platform.

Despite the inevitable gap between the proposed platform architecture and its realization, this work is a proof for the feasibility of a telecommunication platform giving access to diverse network and support to distributed learning environments. Future research could be oriented towards the elaboration of a methodology for the design of distributed virtual laboratories.

References

Anuff, E. (1996). *The Java sourcebook*. Wiley Computer Publishing, 5-28.

Ausserhofer, A. (1999). Web-based teaching and learning: A panacea?, *IEEE Communications Magazine*, *37*(3), 92-96.

Bardout, Y., Hauw, L-H., Pavon, J., & Tomas, J. (1998). CORBA for network and service management in the TINA framework, *IEEE Communications Magazine*, 36(3), 72-79.

Barton, S., Eykholt, J., Faulkner, R., Kleiman, S., Shivalingiah, A., Smith, M., Stein, D., Voll, J., Weeks, M., & WILLIAMS, D. (1992). Beyond multiprocessing…multithreading the SunOS kernel. *Proceedings of the Summer USENIX Conference*, San Antonio, Texas, 11-18.

Berners-Lee, T., Fielding, R., & Frystyck, H. (1996). Hypertext transfer protocol—HTTP/1.0. *RFC 1945*.

Bostica, B., Callegati, F., Casoni, M., & Raffaelli, C. (1999). Packet optical networks for high-speed TCP-IP backbones, *IEEE Communications Magazine*, *37*(1), 124-129.

Collis, B. (1999). Applications of computer communications in education: An overview, *IEEE Communications Magazine*, *37*(3), 82-86.

Collis, B. (1996). Tele-learning in a digital world: The future of distance learning. *International Thomson*.

Kassouf, M., Pierre, S., Levert, C., & Conan, J. (1999). *Modeling a telecommunication platform for remote access to virtual laboratories*, 1999 IEEE Canadian Conference on Electrical and Computer Engineering, Edmonton, Alberta, Canada, 127-132.

Haggerty, P., & Seetharman, K. (1998). The benefits of CORBA-based network management, *Communications of the ACM*, *41*(10), 73-79.

Harkey, D., & Orfali, R. (1998). *Client/server programming with JAVA and CORBA*. (2nd ed.). 3-379. Wiley Computer Publishing.

Henning, M. (1998). Binding, migration, and scalability in CORBA, *Communications of the ACM, 41*(10), 62-71.

Prnjat, O., & Sacks, L.E. (1999). Integrity methodology for interoperable environments, *IEEE Communications Magazine*, *37*(5), 126-132.

Schmidt, D.C. (1998). Evaluating architecture for multithreaded object request brokers, *Communications of the ACM, 41*(10), 54-60.

Seetharman, K. (1998). The CORBA connection*, Communications of The ACM, 41*(10), 34-36.

Siegel, J. (1998). OMG overview: CORBA and the OMA in enterprise computing, *Communications of the ACM*, *41*(10), 37-43.

Vinoski, S. (1998). New features for CORBA 3.0, *Communications of the ACM*, *41*(10) 44-52.