LABVirt – Virtual laboratory for Applied Academic Training

Assist. Codrin NISIOIU, assist. Cătălin SILVESTRU
Department of Economic Informatics, Academy of Economic Studies, Romania

This article presents some aspects about a project integrated in a national research and development project "Information society" (INFOSOC) financed by Romanian Government. A "virtual laboratory" called LABVirt is available for use by Universities, college, high-school. The goal of this paper is to present the architecture of LABVirt and its main components.
Keywords: *virtual laboratory, client-server arhitecture, communication protocol, client applet.*

Introduction

Computer-aided training is an important issue, approached by all academic media. Although experimental education has its definite importance and advantages, *the virtual laboratory* represents an avant-garde concept that takes into consideration teaching aspects within the educational system implemented at all levels.

The Internet-based educational system represents a concept used nationwide in the activity of educational institutions from all levels. The alignment to the latest international trends in the area of computer network organization and functioning based on intranet / internet models (meaning the coming back to the information model in which data and processing is „equally" distributed within the whole network, to architectures with strong servers, using specialized software and hardware solutions which provide information to a wide range of "clients ", that can be both fixed or mobile computers) sets the grounds for closer integration in the information society of the future [1] .

Computer-aided training projects with specific implementations have been developed, with diverse levels of implementation, in European Union countries, predominantly in France, Sweden, Germany, and also in the USA, Japan.

The article represents a synthesis of all features of the *LABVirt* training system. It is structured in three sections as follows: section two presents the architecture of the *LABVirt* system, section three presents the applet-server communication protocol. The final part presents the conclusions reached and a series of directions for further development of the *LABVirt* system.

2. Architecture of the *LabVirt* system

The chosen architecture for implementing the *virtual laboratory* is the client-server type based on two major arguments:

- the processing effort is focused on the server, allowing the clients to run the user application on minimal architectures applications,
- it enables a centralized administration of authorized clients, of laboratory works, as well as that of the whole system..

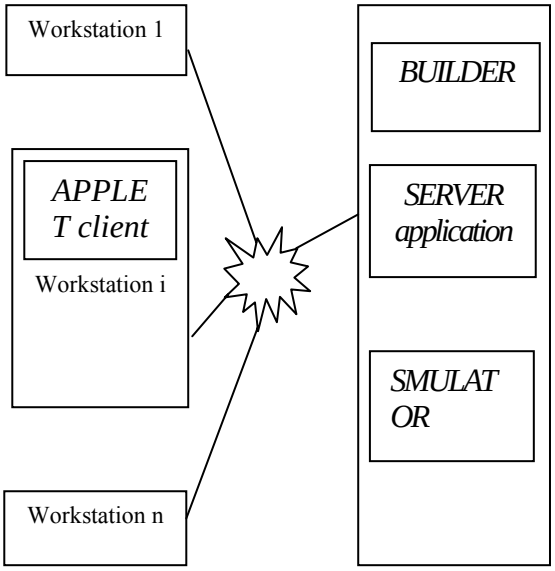

Figure 1. Architecture of the *LabVirt system*

In accordance to aspects already presented in other papers [1, 2], the software components of the system are:

- *the SERVER application* which is the center of the entire *LABVirt* system
- an *APPLET client* (replicated for each user)

- a *SIMULATOR* for general use (MATLAB) that supports the understanding of the functioning of the model, meaning the simulation of the evolution for the system under study.
- a *BUILDER* based on which the files needed for making the *LABVirt* structures and interface are obtained

The *SERVER*, *SIMULATOR* and *BUILDER* applications are on the server, while the *APPLET client* is loaded on the workstation (figure 1) and communicates with the *SERVER* application through the TCP/IP protocole.

In order to access the *LABVirt* services, the clients need to connect via Internet to the *LABVirt* server. In order to be able to conduct laboratory work, the client needs to have an account and password, previously provided, as the server has certain security standards.

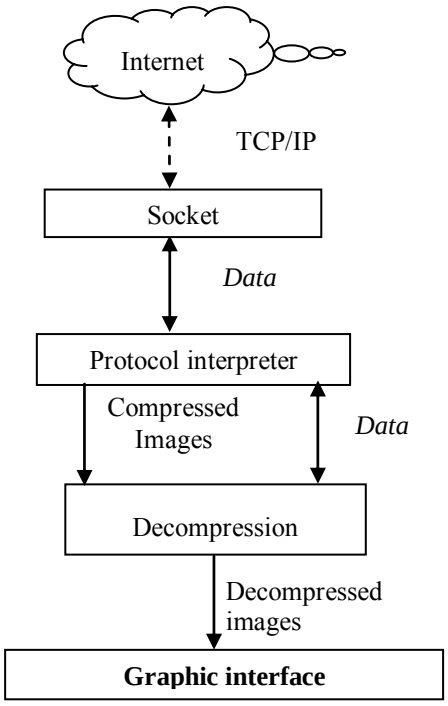

Figure 2. *APPLET client*

Once the password has been accepted by the *SERVER* application, a separate worksesssion is initiated with the client. The connection between clients and the *SERVER* application is done through *threads*, each thread being allocated to a single client.

When ending a thread, previously allocated memory areas are freed dis-allocated and the thread is taken over by collector which sets it ready for connection with new client. (figura 1)

The APPLET client (figure 2) opens a S*ocket* through which data is transmitted towards and from the server by using a TCP / IP protocole. The data communicated can be of the following types:

- text (entry / exit variables);
- binary (images compressed in order to facilitate long-distance transmission)

The data and images received by the server are decodified by teh protocole interpreter, present in the *APPLET client* background. This is capable of separating data from images, by sending the data (exit variables from SIMULATOR) directly to the graphic interface with the user, and the compressed images to the decompression module, which will be sent in final form to the graphic interface.

The values of parameters (insterted by user) need to reach the *SIMULATOR*.

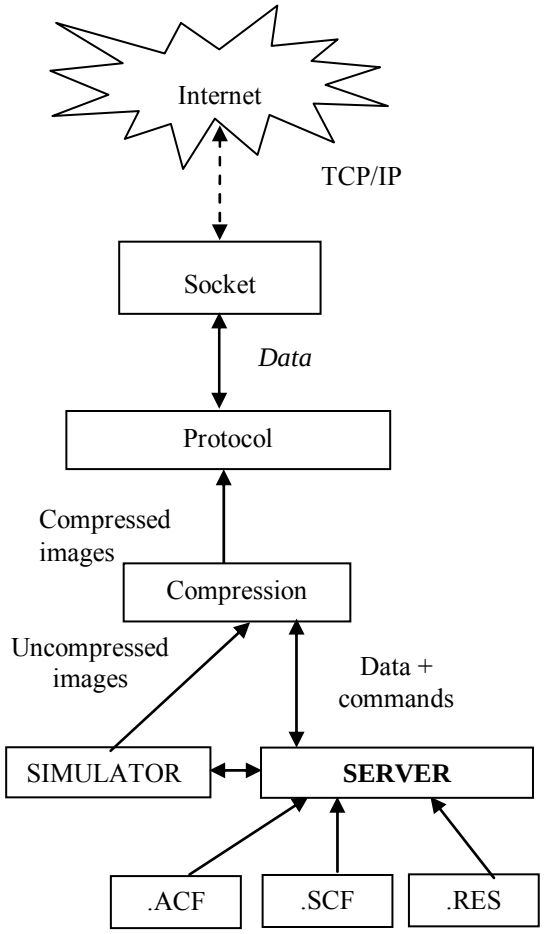

Figure 3. *The SERVER application*

They are transmitted also through the protocole interpreter for administration by the *SERVER application,* that sends them appropriately to the SIMULATOR (figure 3).

The *SIMULATOR* runs on the same computer as the *SERVER application* in order to ease communication and increase data transmission speed between the two entities. The server communicates with the simulator by using a method specific to the simulator: DDE, ActiveX, DCOM, pipes. The *SIMULATOR* maintains separate worksessions for each user. In the case of simulators that cannot support individual work sessions, the separation is done by execution threads. (figure 1).

The administrator board is an interface, component within the *SERVER application* (figura 1) with the help of which

- an administrator is aided in monitoring the functioning of the whole system in view of identifying abnormal functioning;
-different parameters of functioning for the *SERVER application* are modified;
- the number of connections to the system and the addresses from which this has been done are observed;
- changing the status of the *SERVER application* is enabled in view of undertaking needed operations for maintenance and improvements.

3. The applet-server communication protocol

In view of exchanging information between the *APPLET client* and *SERVER,* a communication protocol is required, which to be tolerant to errors that may appear at the TCP / OP connection and also be easy to implement, with the assistance of a logical (functional) communication protocol.

The protocol needs to be able to send and receive two types of data, text and binary formats, in function of the application requirements / needs. The data sent in text format have '\0' as end character, while the binary ones are preceded by information of the other side on the block dimensions that is to be sent. The protocol is made so as to be tolerant to interruptions / discontinuities / idle times that may appear in the communication process.

Protocol structure

Client authentification

First, the user identifies by using the command "AUTH", following which the server will provide confirmation or denial for continuing the session, in function of the validity of the identifier.

File configuration applet request

The APPLET client requests confuguration file for current laboratory work, and will receive confirmation of its existance, the configuration file dimensions and then the file in binary format. In case the laboratory work is not accessible, the server replies with "NOK".

After having developed data structures adequate for the configuration file, the *APPLET client* needs to send a signal to confirm or deny the success of the operation. An interpretation error of the configuration file may appear at the *APPLET client* as result of a variety of reasons. This error must not spread further, as it may make an attempt on the integrity both of the *APPLET client* and the *SERVER application*.

Structure of a module

A virtual laboratory work has several modules (virtual stages) in its structure that are passed through successively as the lab work evolves in the training process. All modules have the same principle of functioning, their structure being identical. Such a module is divided in three stages of communication:

- a request is made for data that represent the objects to be shown to the user
- the transmission of entry data for simulation is made to the server
- data that form the result of the simulation are read from the server.

This structure is repeated for each module separately / specifically until having completed the laboratory work.

The request to activate a simulation module is formulated by the client application by transmitting the command "MOD:xxx". Its receival will impact the selection of data structures regarding the respective module, and automatically the transmission of new

static objects to the client application, so that it may create a new user window suitable for a new simulation session. Each of these objects receives a message of proper receiving, (data communication), and in the end the client sends an OK to signal the server that all necessary data have been transmitted. In case there are errors, most likely of memory allocation, and these objects could not be initialized, the client replies with NOK.

The server gets in stand-by, while the user studies the documentation and prepares entry data for simulation. Once the data are ready, on the user's execution request, the client application sends the command "DATA", which prepares the server for receiving data followed by data transmission, The server answer upon data receival, their validity test and their preparation for the simulation mode is "OK" / "NOK". In case everything went smoothly, the client application gives the "EXEC" simulation execution command and waits for the answer of the server.

In case the simulation takes longer than foreseen, the server sends a presence signal "IN_SESSION".

If the simulation is completed successfully, then the server answers with "OK", and subsequently transmits data in for format mentioned for data communication. If, due to objective reasons, a simulation error occurs, the server promptly replies "NOK", giving the user the opportunity to modify the simulation parameters and try again.

When the user ends the session, the client sends the command „SFMOD"; the server reacts by disconnecting from the current mode and switching to the previous status of command.

Connection termination / ending

Upon completion of laboratory work, the client gives the command „SFLAB", which will precede the connection termination / ending.

Data communication

The protocol needs to support the transfer of two types of data:

- text data using syntax

Object_name:"object_instance"

- binary data :

Object_name: dimension (octet): data_block

4. Conclusions

In this article we intended to highlight the original aspects found in the *LABVirt* pilot product developed as result of the research conducted in the Research Center for Advanced Automatic management of Processes from the Lower Danube University from Galati and the CNCSIS- certified center of excellence from Bucharest Academy of Economic Studies.

The main goal of the pilot software product developed is to provide an alternative to real laboratories, activity which requires time, space and most particularly equipment for adequately undertaking activities programmed in the laboratory curricula. Thus, as result of training, the time spent by students in the laboratory and the number of damage caused to the laboratory works will decrease, while the measurement equipments, the oscilloscopes and the computers will wear down at a slower pace.

In the following version, we shall add the self-evaluation module, currently undergoing testing, and improve the way of transmitting images by implementing compression functions.

Bibliography

[1] Minzu V., Beldiman L., Adrian Serbencu, Adriana Serbencu, Marian BARBU (2003) *Virtual workshop for academic training*

[2] Minzu V and team (2003) *Documentation for INFOSOC contract no 73 / 02.08.2002 – STAGE 1 – chapters 3, 4*

[3] Nisioiu C. (2003) *Developing the client application for the LABVirt application* – graduation project under the supervision of Prof. PhD. Engineer Minzu V

[4] Silvestru C (2003) *Biblioteca digitala pentru informatica economica (digital library for economic informatics)* presented within the national conference "Virtual University in Romania", Bucharest Academy of Economic Studies

[5] Silvestru C. (2005) *E-learning* presented within the International Conference of Economic Informatics – Bucharest Academy of Economic Studies