ARCHITECTURE OF A VIRTUAL AND REMOTE LABORATORY

ALASSANE DIOP
Research Associate, Center for Research LICEF, TELUQ / UQAM, 100, rue Sherbrooke Ouest Montréal
Québec H2X 3P2 Canada
E-mail: diop.alassane@teluq.ca

ABSTRACT

In this paper, we give a detailed description of the Linux e-laboratory (TLL) architecture. To clearly illustrate the various architecture components, we take a close look at various technical solutions so as to describe the approach and methodology used to create the TLL environment for online teaching purposes. To install a solution that meets TLL functional and structural requirements, a five-layered architecture is created, including three layers reserved for servers. This architecture integrates various functionalities using script-based technical solutions, and application and processing server solutions. We have dealt with three points in this paper, namely: the architecture of the Linux e-laboratory (TLL), technical solutions used to install a TLL environment and functionalities for interlinking the various model components.

Keywords: Remote Lab, Distance Learning, E-Learning, Linux, Networks

1. INTRODUCTION

The e-learning has become a very widespread business unit in the academic and professional training. It showed an exponential development as well as conceptual than technical. Our research demonstrates how to use the concepts and the techniques developed in the field of e-learning in order to result in an architecture concerning the laboratory on line of telematics, which can serve as a base of the realization of a switchboard. The developed and suggested solutions are based on three main problems that are recurring at the time of real laboratory on line installing. The research work focalizes chiefly on these problems in order to propose a solution for the three questions it raises. (i) How to put up a laboratory on line for practical work, while keeping the practical and the real aspect about the proposed experiences. (ii) What are the possibilities offered by the e-learning in the case of installing an environment of flexible, easy use and that can be integrated. (iii) What are the contributions of our proposition on e-laboratory in comparison with the usual laboratories of telematics. To solve these problems, a practical example is developed for the implementation for an ideal structure for a laboratory on line. This example proposes new solutions for the self-display of an operating system on a virgin computer (a computer with no operating system). This solution uses an environment of distance learning based on existing network applications (Telnet, SSH…) and on new applications developed for the needs of this study. The same solution will also allow the configuration of network computers in an apprenticeship environment. It integrates a tool of management of activity in use of the laboratory by a system of reservation. The latter permits to share the resources between the different users of the laboratory on line. Therefore, the flexibility of the integration and the suppleness of such a solution are demonstrated. For its realization, a proposition has been made on the way of integrating the Linux e-laboratory in different platform of e-learning like WebCT, Claroline and Ganhesa.

2. LINUX E-LABORATORY ARCHITECTURE (TLL)

This section covers relatively technical content. But we’d like to point out that the concepts developed here are easy to implement on a real device.

An e-laboratory application implies that at least four significant architecture components are in place: the client navigator, Web server, application server and database server.

Key architecture components are located on the LMS “Learning Management System” server. In many cases this architecture is in fact that of a
standard Web application. In order to achieve a high-performance operational functioning of the system, a five-layered model with four main functionalities is created. Each layer component plays an interlinking and/or processing role while connections and interactions are being established between the learner workstation and target equipment in the real laboratory. (See FIGURE 1)

The Learner Workstation: This is a standard Web (HTML) navigator that must provide a fill-out form support. Thus, it’s a tool that’s commonly used. The page contains a completely formatted interface, presented by the navigator in the learner’s window. The LMS Server: This is a Web server that acts as the main access point for all learners. In response to specific inquiries, the Web server will trigger processing from the DB “Database” server. In all cases, the result is an HTML page that can be displayed by the learner’s standard HTML navigator. Inquiries are pages generated on the LMS servers in the form of PHP script pages. These pages provide access to all resources found on the LMS server.

The DB Server: It serves to maintain persistence in the relational database. The simplest way to connect to the system is to allow for direct access to PHP script pages in the persistence component. This direct access requires using a standard library source code to access data, such ODBC “Open Database Connectivity” or JDBC “Java Database Connectivity”. For sturdier or more complex systems like TLL, we have developed and included an object layer that takes care of extracting data from the database.

The Application Server: It’s made up of executable applications that allow for extracting data and executing specific programs for the TLL.

Target Laboratory Equipment: This includes machines, hubs, routers and other types of active components. This equipment can be directly manipulated by the learner via the LMS server, the DB server and the application server. All three servers are installed as client/server applications, which facilitates intercommunication among the servers. Figure 1 illustrates the TLL architecture with the various protocols, applications and servers required for the environment to function.

To create and install such an architecture, one can use one of several application development technologies. To develop TLL, we chose the PHP technology for the Interface component, and C++ and Java solutions to program client/server applications on the various layers of the model.

PHP is used to manage the TLL user interface, which makes for a light client. All programs developed in PHP are executed on the LMS server. The LMS server communicates via an HTTP protocol with the navigator used by the learner. Along with this communication protocol, the LMS server uses a built-in functionality in the application set up in this server, described in Section 3. The architecture illustrated in Figure 1 demonstrates the intercommunication among the various actors involved in TLL. Communication takes place in the form of client/server applications that take care of executing the various tasks for the good functioning of the system. These different server applications are developed in C++ and Java languages. The architecture is very modular: applications “run” independently of one another, but are interlinked through the communication infrastructure of our model. This makes for smooth management and one can displace an application from one server to another. In TLL, the C++ language is used to develop deployment programs based on SHELL commands executed in Linux machines. For remote machine emulation programs, the Java language is used for real time. The Java programs are combined with the VNC (Virtual Network Computing) [7].

3. COMPONENT INTERCOMMUNICATION FUNCTIONALITIES

FIGURE 2 illustrates the communications mechanism linking the components used to set up the TLL environment.

When a learner connects to TLL, his/her data are sent to the LMS server via an HTTP protocol. The LMS server creates a session based on the learner’s “user name” and “password”, the “IP address” of the machine being used, the “date” and hour of the “connection”. Throughout the connection time, the session identifier is used to locate and identify the learner. The server actors are thus executed as client/server applications based on four key functions: “TLL_SEND, TLL_RECV, TLL_FRMT, TLL_EXTR”.

The “TLL_SEND” Function: With this function, one can send information in text format in terms of an identifier specified in the inquiry heading. To send data, the actor sends the following instruction:

```
TLL_SEND
<IP_SRC_ADDRESS> <IP_DEST_ADDRESS> <DATA> <FORMAT> <PRIORITY>
```

“IP_SRC_ADDRESS”: This lets one determine the IP address of the machine that sent the
“Machine’s source address or machine’s name” data. IP_DEST_ADDRESS: This lets one determine the IP address of the machine where the “Machine’s destination address” data were sent. “DATA”: This is a set of “ASCII Code” alphanumeric characters representing the information to be sent. “FORMAT”: This lets one determine the sent data format; “VAR, TXT, SQL, REMBO, HTML”. “PRIORITY”: The latter parameter is optional, if it’s rated as –P, the message is low priority and if it’s rated as +P, the message is high priority. If no priority is specified, the system handles the message in normal mode.

The “TLL_RECV” Function: With this function, one can receive information in terms of function parameters. The function syntax is as follows:

```
TLL_RECV
 <IP_SRC_ADDRESS>  <IP_DEST_ADDRESS>  <DATA>  <FORMAT>  <PRIORITY>
```

The “TLL_FRMT” Function: With this function, one can prepare data received in terms of a given outgoing format. This function uses information received through the TLL_RECV function whose syntax is as follows:

```
TLL_FRMT
 <DATA_IN>  <FORMAT_IN>  <DATA_OUT>  <FORMAT_OUT>
```

“DATA_IN”: These are the raw data received. “FORMAT_IN”: This lets one determine the received data format. “DATA_OUT”: These are the data extracted from the raw data. “FORMAT_OUT”: This lets one determine the extracted data format.

The “TLL_EXTR” Function: With this function, one can extract or execute a command in terms of a given application. Function syntax is as follows:

```
TLL_EXTR
 <DATA_IN>  <FORMAT_IN>  <PROGRAM>  <DATA_OUT>  <FORMAT_OUT>
```

“PROGRAM”: This is the program used to execute the command or to extract information.

4. PHYSICAL MODEL

The physical model of data is made up of 3 tables. The "STUDENTS" table which makes it possible to store all information concerning the students, this information is used to identify the student at the time of the phase of authentication. The "EQUIPMENT", "VIRTUAL MACHINE", "SYSTEM" tables makes it possible to keep all information concerning the computer where the operating system is installed by the student: creation date, name given to the computer and the identifier of the student who did the installation. The "CONFIGURATION" table makes it possible to store the configuration made on the computer. This information is represented as a file which is the image of the installation made by the student on the computer. The model is represented in the following way (See FIGURE 3).

A) MANAGEMENT RULES

- A virtual machine belongs to only one student
- A student can have several virtual machines
- An image is only for one virtual machine
- A virtual machine can have several images

B) PROPOSAL FOR A SCENARIO

- Connection, Authentication and Authorization Identification (AAI)
- Looking for available remote computer of the student
- Choice of a virtual machine (If the student does not have any virtual computer, the system creates a virtual empties computer)
- Looking for of the various images of the selected virtual computer
- Choice of an image
- Restitution of the image on the level of the virtual computer
- Emulation of the computer

5. CONCLUSION

One of our targeted goals was to design distributed online laboratories, namely laboratory facilities that are accessible from several different sites. With the proposed architecture, one can centralize the teaching platform, thus providing the learner with a unique environment. From a logistic point of view, being thus able to carry out laboratory experiments from anywhere and at any time makes it possible to keep experiment facility and management costs at a minimum, irrespective of an increase in the number of students. This
solution eliminates the need to move heavy or cumbersome hardware in a classroom to carry out laboratory experiments or demonstrations. With such a solution, one can also share resources and help emergent countries benefit from expensive resources to which they would otherwise not have access. Dividing the model into several layers makes for simpler learner/machine interactions. Using light clients without having to install applications on the learner workstation results in a leaner and more straightforward architecture. This also makes for easy user access, since users can connect to the system at any time and anywhere, wherever there’s a machine equipped with an Internet connection, to run experiments or to administer a remote online system.

6. REFERENCES

[1]. VITELS, 2000, Virtual Internet and Telecommunications Laboratory of Switzerland (VITELS), http://www.vitels.ch.
[3]. SVC, Swiss Virtual Campus, 2000. Program launched by the Swiss Confederation to promote new communication and information technologies in higher education, http://www.swissvirtualcampus.ch
FIGURE 1: PROTOCOLS AND APPLICATIONS USED IN TLL

FIGURE 2: METHODS OF COMMUNICATION OF SYSTEM FUNCTIONS

FIGURE 3: DIAGRAM OF THE DATA BASE