CONTRIBUTION TO THE DEFINITION OF BEST PRACTICES FOR THE IMPLEMENTATION OF REMOTE EXPERIMENTATION SOLUTIONS

Xavier Vilalta, Denis Gillet and Christophe Salzmann

EPFL, CH - 1015 Lausanne, Switzerland

Abstract: This contribution presents a general framework for implementing and deploying remote experimentation solutions. Along with the described guidelines and recommendations, a real online experiment is depicted. In addition, different solutions for the development of the necessary software tools are analyzed, and a comparison is given to help on the choice of an optimal solution when implementing these real-time internet services.

Keywords: Automatic control, Remote Experimentation, Flexible Learning, Real-Time Internet Services.

1. INTRODUCTION

Hands-on laboratory practice has always been an essential ingredient to sustain the learning activities in engineering education. It is recognized as an efficient approach for students to effectively assimilate knowledge and to develop a professional approach to solve real-world problems. Since laboratory exercises are often a combination of individual preparatory activities followed by experimentation and analysis stages carried out in team, it is also valuable for reinforcing both the students' autonomy and their teamwork skills.

Around 1995, with the general acceptance of the Internet as the common worldwide communication channel and the migration of most control devices instrumentation to personal computers, many trials for remote monitoring and for the control of laboratory equipment have been carried out in academic institutions. Nowadays, this paradigm designated as remote experimentation is becoming a key feature for deploying distance and flexible engineering education. In addition to the removal of time and location constraints for students access, the key advantage of the remote experimentation paradigm is the possibility for educational institutions to share costly laboratory resources. Consequently, various distributed laboratory initiatives have been launched recently, one of them being a distributed laboratory composed by a network of experiments that are made accessible to an authorized group of teachers and students (Schmid *et al.*, 2001).

Because of the etherogenous nature of laboratory equipment, it is almost impossible to standardized the implementation of remote experimentation solutions. However, it is essential to define some best practices for the implementation of new remotely accessible laboratory setups or for the upgrade of existing ones. It is obvious that the students' benefit of using distibuted laboratory resources will be lost by a cognitive overload if they get completely new interfaces and behaviors when they access different experiments.

Best practices can be defined for the selection of the physical systems in automatic control education as detailed in Section 2, at the level of the client-server architecture as illustrated in Section 3, at the level of

the user interface to provide a seamless look and feel as described in Section 4, and finally at the level of the communication layers implementation to provide the best possible level of interaction as shown in Section 5.

2. PHYSICAL SYSTEM SPECIFICATIONS

The introduction of physical equipment as didactic resource is the fundamental principle of laboratory experimentation. The nature of the equipment is strongly related with the pedagogical goal of the laboratory sessions. We can distinguish three different objectives that can be targeted using physical equipment:

- Illustration of a concept that has been presented in a textbook or taught in the classroom (knowledge). In such case, the equipment is introduced to sustain the concept appropriation according to its relevant physical peculiarities.
- Development of a pertinent methodology for scientific experimentation or for technical implementation (general know-how). In such case, the equipment is introduced as a media according to the spectrum of activities that it helps to cover.
- Acquisition of operational skills on professional equipment (specific know-how). In such case, the equipment is simply the one that has to be mastered.

There is only one latitude in the choice of the equipment for the two first objectives. Balchen has summarized the characteristic that such equipment has to exhibit (Balchen *et al.*, 1981). It has to be:

- Relevant with the pedagogical objectives.
- Realistic compared to its possible industrial counterpart but simple enough to make it understandable in a limited time.
- Adequate from a dynamical point of view. That is fast enough to make it interesting but slow enough to make it observable using typical Web cameras.
- Multimodal. That is addressing student perception at different levels (visual, auditive, haptic, …).
- Safe in itself as well as for the operator.

Wellstead has proposed in addition to the above mentioned characteristics (Wellstead, 1990) to chose scale-model systems. His reasons are similar to the previously ones. As a matter of fact, the time constant of dynamic system is often inversely proportional to its size. Usually, smaller systems present less danger and require less power.

The mentioned characteristics are applicable either for local or remote experimentation. When a remote access is provided to the laboratory equipment, it is required that the physical system and its instruments are completely operable using a computer. This also means that it has to have all the necessary sensors and actuators. Furthermore, advanced perception solutions have to be introduced to compensate for the remoteness.

The EPFL's main interest is to implement remote experimentation setups for educational purpose with applications for the control of mecatronic systems. Such systems have time constants in the same order of magnitude that the delay usually introduced by the Internet communication. Therefore, the main constraint is to reproduce the dynamical behaviors of the distant equipment and to provide adequate sensorimotor functionalities to enable effective interaction (Gillet *et al.*, 2001a).

3. CLIENT-SERVER ARCHITECTURE

In order to illustrate the components that are necessary to implement the client-server architecture dedicated to remote experimentation, a real example is introduced. This example is the implementation of the online Internet access of an inverted pendulum located at the EPFL (Gillet *et al.*, 2001b).

The proposed solution is a three-layered architecture (also called three-tier model) which is build up around the following main components: an experiment server, an administrative server and a remote client as can be seen on Figure 1.

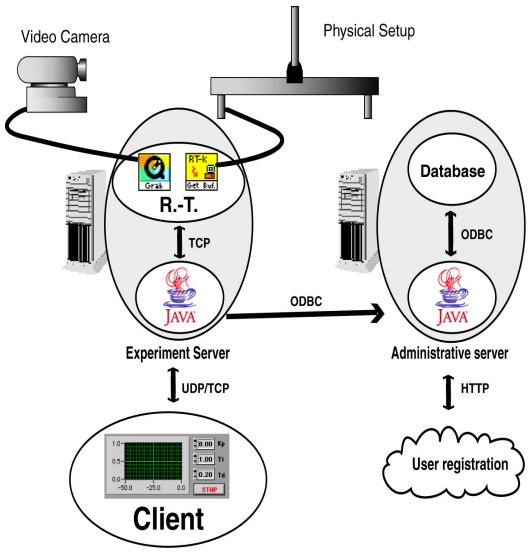

Figure 1. Client-server architecture. This figure depicts the experiment server that controls the experiment, the administrative server containing the business-logic components and the client software.

The experiment server includes a real-time kernel that executes the local control of the real system, I/O peripherals to interface the experiment with the equipment, sensing peripherals such as microphones and video cameras, the communication layer to

broadcast the collected information to remote clients and the Java glue for users' validation.

The administrative server contains a database that stores all the necessary information about the users and the necessary Java glue to provide this information to the experiment server. If a user registration is possible online, using a web browser for example, another Java component has be added. The checking of user privileges is only necessary at the beginning of and at the end of a remote experimentation session, respectively to set up the communication parameters and to release the used resources.

According to this structure, the last component of the system is the client, the software that remote users employ to connect and control the distant setup and to display the received information. The feedback provided to the client graphical user interface (GUI) shows the effects of the actions taken by the user on the experiment. It can contain real-time video and audio information, as well as the values of relevant signals read on the experiment. The received information is displayed in a cockpit like GUI as detailed in the next section.

4. COCKPIT-LIKE GRAPHICAL USER INTERFACE

Carrying out a remote experiment is a matter of observing and acting on a system that is located at distance using convenient sensors and actuators. Hence, end users in remote experimentation can be seen as pilots acting in airplanes. They observe the reality through the plane windshield (video image) and through the instruments (scope window) and they act (pilot) the plane using a joystick and other means. To take advantage of this similarity, a general cockpit metaphor is chosen to design the graphical user interface (GUI) for the remote experimentation client software.

In a remote experimentation cockpit, all the relevant information necessary to grasp the experiment behaviors (the environment state and the controller settings) is integrated on a single screen. The following parts (or components) can be distinguished on the user interface (as seen on Figure 2):

- An administration part allows the user to login/ logout and to adjust some software settings (such as video refresh rate, image size, sound level …).
- A perception part including the real-time display of video images and, if implemented, the representation of virtual or augmented reality views showing the remote experiment and its surrounding environment.
- An operation part that lets the user modify the experiment settings to change the behavior of the system under experimentation, the operating conditions or the instruments configuration.
- A data management part that enables information storage and retrieval.

These parts could appear on the screen alone or grouped into a bigger component, in order to enhance the clarity of the user interface. However, cognitive considerations lead to a user interface design as versatile as possible. The key point is to let the users choose and organize as much as possible the different parts as they deem it necessary, while providing default configurations to ease the comprehension of the basic client-software functionalities.

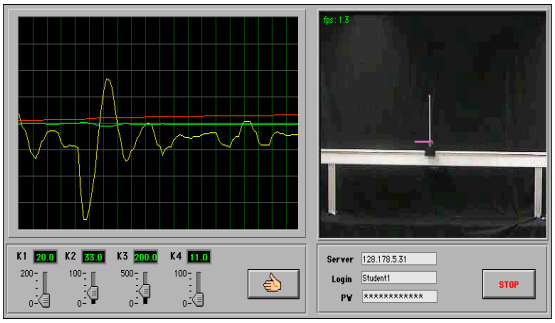

Figure 2. Screenshot of the LabVIEW inverted pendulum client software.

The details of the user interface parts are given below. These details have to be seen as good practices more that as definitive recommendations.

The administrative part. It has to be kept as reduced as possible. Most of the administrative settings have to be chosen at a stage that occurs before entering the cockpit. The administrative settings that are parts of the cockpit are mainly remainders of key information and elements required to configure the general mode of operation. The typical information of relevance for remote experimentation includes:

- An identifier of the operated experiment (such as its IP number).
- A user login/logout mean (usually fields to provide username and password, and a login/logout button). For such an access button, it is useful to use icons similar to the ones founded on a real life equipment, such as cell phones, since it is a matter of opening/ closing a communication channel.
- An operation mode indicator/selector, to specify modes such as administrator, observer (if looking at the experiment carry out by another user) or master (if being the one that as the control of the experiment), on queue (if waiting for the access) or on line (if having the access), default or demo (to limit inappropriate operations in certain circumstances).
- An experiment start/stop mean (once connected, it has to be possible to start and stop the experiment, which is different of being logged in). For such

action buttons, it is useful to use icons similar to the ones funded on remote control devices for audio/ video systems.

- A reset button that brings the experiment back into a safe (initial) state.
- A language selection (local language + at least English).
- A surrounding environment setting (light on/off, power supplies on/off, …).
- Links to extra features or resources, such as online help and support, documents related with the experiment or collaboration tools.

The perception part. Perception is a difficult challenge when considering providing the necessary information to catch the changes and the dynamical evolution of a remote physical experiment. No single tool can bring a solution, it is necessary to combine various representation means to provide information on both the internal state and the external conditions. It is also necessary to introduced advanced solutions to compensate for the remoteness from a telepresence point of view (Schmid *et al*., 2001) (that is providing the feeling on being in the real laboratory) as well as for an interaction point of view (that is avoiding as much as possible the effect of transmission delay). Typical components that enhance the level of perception are:

- Scope for versatile selection and visualization of measurement or internal signals.
- Real-time video views of the remote equipment and its surrounding environment, including the necessary zooming and orientation mechanisms to control the cameras or any other sensing devices.
- Audio feedback (oppositely to video-conferencing, audio has a lower priority than the video in remote experimentation).
- Switch to commute between the real experimentation and the simulation mode (if available). Built-in simulation capabilities are convenient for off-line tuning and training, on-line validation, fault detection and data recovery, as well as for augmented reality (discussed below). Simulation requires a mathematical model of the experiment, which is something difficult or too complex to create/develop. When using simulation the additional difficulty of updating the model parameters and the initial simulation conditions arise.
- Virtual reality representations of the remote equipment and of its surrounding environment. The virtual representations can be either driven by real measurements or simulated data (if built-in simulation capabilities are available). To create an augmented view, virtual reality representations can be combined with video views. This required the implementation of mechanisms to synchronize real and virtual views both spatially and temporally.
- Additional sensorimotor functionalities such as force feedback produced by haptic devices. Such an addition can be highly valuable but is usually inaccessible for most of the remote users.
- A progress bar displaying the remaining connecting time or the elapse time for batch experiment.
- A replay button to show logged data or store video sequence of the experiment.

It is not mandatory to combine all these components together, but at least video or virtual reality should to be available.

The operation part. Compared to the administrative settings, the operation settings are the ones that interactively change to sustain the students learning activities. Typical settings are:

- Various control algorithms and structures.
- Operating points and reference trajectories.
- Controller parameters, including the sampling period.
- Remote perturbation functionality (to remotely apply perturbations while the operations are running). This functionality is essential in automatic control education to validate the achieved controller performances.

The data management part. Data management is a key feature to sustain the learning process. Such a component has to be design to provide ways to save and restore the individual experimentation context, and to allow later exploitation of the experiment results. From basic to advanced data management features, ones can mention:

- Printing and saving (screen shots or data with decimation option).
- Continuous data logging (with decimation option).
- A save/restore context functionality (to save the current settings of the experiment for a later access or to bring the experiment in a previous state in order to resume an interrupted activity).

It is important to mention that not only data have to be logged, but also the experiment configuration such as the state of the experiment and the control device settings.

5. COMMUNICATION LAYER AND CLIENT SOFTWARE IMPLEMENTATION

5.1 Communication layer.

The IP protocols (IP, TCP and UDP) are the de facto protocols used for Internet communications and should be chosen as the core tools to build the communication layer to support remote experimentation. There are two types of information

transmission between the server and the client. The first type of transmission, used to setup the communication, should use a reliable transport protocol (TCP). The second type of transmission, containing time-critical information exchanged between the experiment server and the client, should use an efficient transport protocol (UDP). This permits a fine control of the interaction, as well as the implementation of advanced adaptation schemes to compensate for the Internet bandwidth variation.

The communication between the server and the client is composed by the following stages: Negotiation, operation and termination. The negotiation stage (also called login) requires various data to be transmitted in a secure manner. For this purpose, the Real Time Streaming Protocol (RTSP) has been designed (Schulzrinne *et al*, 1998). The benefits of using a RTSP-like protocol is that the system will follow a standard method to negotiate and establish the connection and to identify the user. For typical needs the full RTSP implementation might not be necessary. Once the user access has been granted, the operation stage begins. On the top of the IP protocol the RTP/RTCP protocol (or a subset of it) carries the information between the server and the client and vice-versa (Schulzrinne *et al*, 1996). The RTCP protocol provides the state of the link estimated by the client. Based on the client information, the server makes the necessary adjustments to adapt the flow of information to the available bandwidth. Finally, at the termination stage, the client sends a logout command to the server to close the connection and to free used resources, by using again an RTSP-like protocol.

5.2 Client software implementation.

When implementing client software, it is important to take into consideration the multiplatform nature of the computer resources own by students, especially with the emergence of Linux. To provide solutions as open as possible, LabVIEW and Java implementations can be considered as good alternatives.

LabVIEW is a proprietary software, but it is possible to use it as a development environment to compile stand-alone applications for various target platforms. Such instrumentation-oriented applications can be used freely for educational purposes. Moreover, it is also possible to sign Virtual Instruments (LabVIEW codes) that can be executed using the LabVIEW player. This player is freely available on various platforms and operates in the same manner as the well know RealPlayer from RealNetworks.

Java applications can take different forms. For example, it is possible to run a Java application in a Java Virtual Machine (JVM) embedded within a Web browser (Applet) or into a JVM running directly on top of the OS (Application). These approaches which are compatible with the previously mentioned recommendations, have benefits and limitations. Their peculiarities for the implementation of remote experimentation solutions are described in Table 1.

Table 1. Client software implementation peculiarities.

	LabVIEW	Application	Applet
Platform	Windows, MacOS UNIX/ Linux	Windows, MacOS, UNIX/ Linux	Windows, MacOS UNIX/ Linux
Performance	High	Medium	Medium
Deployment	Easy	OS dependant	Very Easy
Security	Medium	Medium	High
Development	Proprietary	Broad range	Broad range
Reusability	High	High	High

For further analysis it is interesting to take into consideration the remarks given below.

Platform. The LabVIEW distribution depends on the availability of either the application builder or the player relevant for the targeted platforms. The possibility of running the Java applet is dependent on the existence of a Java-compliant browser for the desired platform. The version of the JVM is also an important aspect to ensure a smooth, trouble-free execution of the Java solutions.

Performance. Currently, LabVIEW applications are more efficient than the Java solutions. This difference is quite significant when extensive decoding is necessary to render the video images.

Deployment. LabVIEW stand-alone client software needs no additional tools to be executed. As a counterpart, they are quite large to download. When running Java client software, it is necessary to take into account the fact that different systems can have different requirements for the launching. Some commercial tools, such InstallAnywhere can be used to help in the software deployment by providing installers for all the platforms. Moreover, it could be necessary to have special OS access rights to install the necessary components to run the application. Such a situation can be encounter when using personal computers on campus. In this case, it is preferable to provide client software as applets.

Security. There is no way to restrict the system access to a stand-alone LabVIEW application. To increase

the degree of trust and the certification of the origin of the code, signed virtual instruments can be used. Java also provide such signature capabilities in addition to the security implemented at the JVM level. When running an applet, the security is more a browser-dependent point. There are some browsers which have a finer tuning system for security, where the user can manually choose the applet privileges. In other browsers the security model allows only the user to grant or deny a group of actions to the applet, restricting the flexibility of the choice.

Development. There are many tools for Java development, ranging from the simpler Java Development Kit, freely available from Sun, to many commercial Integrated Development Environments for a broad range of platforms.

Reusability. In LabVIEW, the source code is common to all platforms, but a compiled version is needed for each platforms/ OS. Using Java, different browsers or different versions of the same browser can have slightly different Java implementations that could make the environment incompatible with the applet, requiring accurate programation or the development of browser-adapted solutions, such as code that detects the browser vendor and version and that runs the apropiated applet for this precise setup.

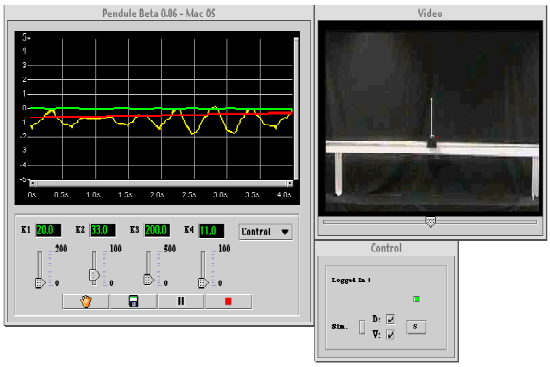

Figure 3. Screenshot of the inverted pendulum client software implemented as a Java application.

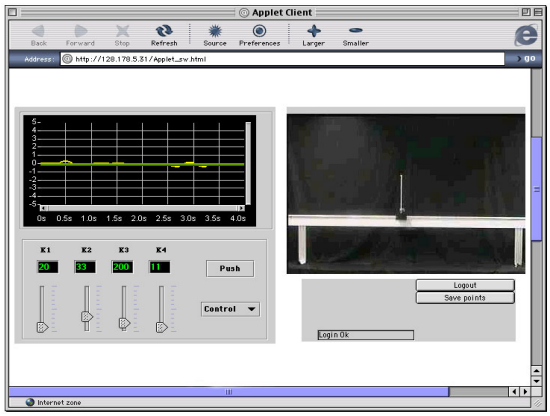

Figure 4. Screenshot of the inverted pendulum client software implemented as a Java applet.

The aspect of the cockpit-like user interface can be seen on Figure 3 for the Java application and on Figure 4 for the Java applet.

6. CONCLUSION

This paper describes guidelines for implementing remote experimentation solutions, covering the physical system specifications, the client-server architecture and the user interface. Following this guidelines, the advantages of an open and multiplatform solution based on Internet standards is shown, and the comparison between three different software approaches covers a broad range of cases with diverse requirements. Java shows promising cross platform solutions and security management. However for highly demanding applications, LabVIEW compiled code is more efficient and should be preferred.

ACKNOWLEDGMENTS

This work was partially supported by the Swiss National Science Foundation, grant #5003-045347 as part of the Swiss Priority Program for Information and Communications Structures and by the EU IST Programme under project contract IST-1999-20827.

REFERENCES

Balchen, J.G., M. Handlykken and A. Tysso (1981). The Need for better Laboratory Experiments in Control Engineering Education. P*roc. 8th Triennial IFAC World Congress,* **7**, Kyoto.

Gillet, D., H.A. Latchman, Ch. Salzmann and O.D. Crisalle (2001a). Hands-On Laboratory Experiments in Flexible and Distance Learning. *Journal of Engineering Education,* April, pp. 187-191.

Gillet, D., Ch. Salzmann and P. Huguenin (2001b). A Distributed Architecture for Teleoperation over the Internet with Application to the Remote Control of an Inverted Pendulum. *Lecture Notes in Control and Information Sciences 258: Nonlinear Control in the year 2000"*, **1**, pp. 399-407, Springer-Verlag, London.

Schmid, Ch., T.I. Eikaas, B. Foss and D. Gillet (2001). A Remote Laboratory Experimentation Network. *1st IFAC Conference on Telematics Applications in Automation and Robotics*, Weingarten, July 24-26.

Schulzrinne, H., S. Casner, R. Frederick and V. Jacobson (1996). RTP: A Transport Protocol for Real-Time Applications. *RFC 1889*, January.

Schulzrinne, H., A. Rao and R. Lanphier (1998). Real Time Streaming Protocol (RTSP). *RFC 2326*, April.

Wellstead, P.E. (1990). Teaching Control with Laboratory Scale Models. *IEEE Transactions on Education,* **33**, Number 3.