Requirements for Real-Time Laboratory Experimentation Over the Internet

Ch. Salzmann[1], H. A. Latchman[1], D. Gillet[2], and O. D. Crisalle[3]

[1]Electrical and Computer Engineering Department
[3]Chemical Engineering Department
University of Florida
Gainesville, Florida 32611-6005 USA
csalzman@cise.ufl.edu
latchman@list.ufl.edu
crisalle@che.ufl.edu
tel. (352) 392.49.50
fax. (352) 392.00.44

[2]Institut d'Automatique
École Polytechnique Fédérale de Lausanne
(Swiss Federal Institute of Technology)
CH-1015 Lausanne Switzerland
Denis.gillet@epfl.ch
tel. (+41) (021) 693.51.68
fax. (+41) (021) 693.25.74

Abstract

A prototype system based on an inverted pendulum is used to study the Quality of Service and discuss requirements of remote-experimentation systems utilized for carrying out control engineering experiments over the Internet. This class of applications involves the transmission over the network of a variety of data types with their own peculiar Quality of Service requirement. These data types include video and audio images from the process environment, signal traces related to the acquired measurements, control instructions sent to the process actuators, and other information concerning the states of the process. The set up includes a physical system (an inverted pendulum) as well as a local server fitted with a video camera, data acquisition boards, and network connectivity that permits interactions with remotely located clients. The paper discusses relevant issues of the design, and presents a paradigm for operation based on a client-server configuration and a master-client mode of service. The information streams involved in the process are classified in four groups of different transmission priorities, namely, a parameter stream, a data stream, an administrative stream, and an audio/video stream. The paper analyzes the performance and requirements of the system based on the results of transatlantic tests. The results of the analysis show that real-time remote-control experimentation over the Internet is in fact a new kind of network application featuring its own requirements that are different from those of related technologies used for video-conferencing/broadcasting. In order to overcome the current lack of predictability of the Internet, the final section of the paper suggests improvements such as adapting the priority of the different streams to the Internet bandwidth and to the user's needs.

1. Introduction

The Internet and modern multimedia tools make it possible for students to access a university lecture from remote locations such as the home of the workplace. In traditional engineering classes, concepts taught through lectures are often reinforced by practical experimentation carried out in laboratories sessions that are attended by the students at the physical site of the facilities. A new paradigm to make the experimental activities available to remotely located students has been developed at the Swiss Federal Institute of Technology in Lausanne, Switzerland, and tested within the university's LAN. An extension of the set-up has now been introduced to reach more distant

locations, allowing the sharing of the laboratory facilities with remote universities.

The development of remote-experimentation facilities is motivated by the fact that presently, as never before, the demand for access to the laboratory facilities [1] is growing rapidly in all engineering colleges. At the same time the number of students is increasing while the allocated laboratory resources do not keep up with the pace. Being able to make the laboratory infrastructure accessible as virtual laboratories, available 24 hours a day and 7 days a week, go a long way towards addressing these difficulties, and would also contribute to lower the costs of operating the laboratory in the long term. This increased availability would be obtained by allowing students to reach the laboratory facilities via modem from home, or from other points of network access, such as computers available at different campus locations [2]. Furthermore, availability of such facilities would enable participation in the laboratory experiences to students who are remotely located, such as practicing engineers who would participate using computers located at the site of their employers or at their own homes.

This paper describes a remote-experimentation set up used for carrying out laboratory exercises in control engineering, and quantifies observations gathered from transatlantic experiments between the Swiss Federal Institute of Technology in Lausanne, Switzerland, and the University of Florida in Gainesville, Florida. Section 2 presents an overview of the systems used in the laboratories of the Institute d'Automatique of the Swiss Federal Institute of Technology to support distance experimentation, and describes the basic components of the system. Section 3 addresses basic issues that are of importance to the effective design of a remote experimentation system. Section 4 describes the operational paradigm which has been developed, including the client-server configuration, the management of requests placed by clients, the classification of the information streams transmitted, the relative hierarchy of the processes involved as well as other relevant requirements. Section 5 presents a contrasting discussion between the requirements of real-time experimentation and conventional videoconferencing/broadcasting systems. Finally, Section 6 discusses some solutions for optimizing the use of the available bandwidth. Final remarks are given in Section 6.

2. Overview of the Remote-Experimentation Set Up

This section describes typical processes used for remote laboratory experiments, and discusses the basic components of a prototype set-up.

2.1. Typical processes used for remote laboratory experimentation

Many mechatronic systems— *i.e.*, those that combine electrical and mechanical parts— used in a control engineering laboratory are well suited for remote experimentation. They are attractive to the students because they often yield responses that are easy to identify visually. Furthermore, experimentation can typically be conducted in a reasonable amount of time. For example, a complete laboratory experiment could take between one and two hours of work, during which period the student carries out modeling and design studies, including shorter periods (say, 5 to 15 minutes) of interaction in real-time mode with the experiment for measurement and control purposes. This paper focuses on an experiment based on an inverted pendulum available at the Swiss Federal Institute of Technology. For contextual reference, we note that two other systems, namely an helicopter and an electrical drive [3], are also accessible via the Internet.

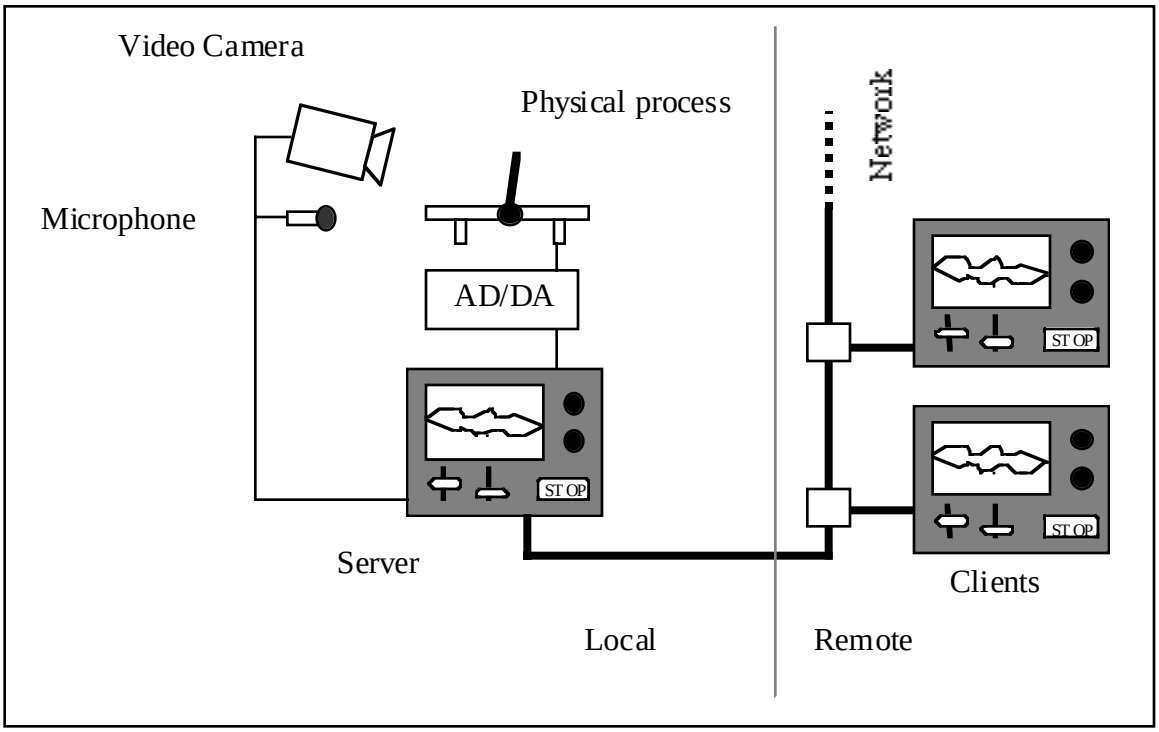

Figure 1. A physical system (inverted pendulum) communicates with a local server via AD/DA cards, and a network connection gives access to multiple remote clients.

2.2. Basic components of the remote-experimentation system

The configuration of the remote-experimentation system is shown in Figure 1, where the pendulum system is shown along with the DA/DA cards that connect it to a local server. The server in turn displays all relevant signals to a number of remote clients that access the local systems via the Internet. The server also receives commands from selected remote clients, and implements the commands locally on the physical system. The server and the computers platforms for the clients have identical display screens, thus giving the remote users the opportunity to interact with the physical system in a fashion analogous to the way local users do. The network transmits input and output data streams as well as audio and video information.

The user can interact in real-time with the experiment through a graphical user interface (GUI) built using the LabVIEW graphical programming language [4]. This interface (Figure 2) has been designed to be the same for a local or a remote experimentation. It is composed of an oscilloscope window where the measurements done on the real process (position, angle, etc.) are displayed. Four sliders represent the parameters provided by the user to the controller. Local users have the benefit of being able to introduce perturbations by making physical contact with the pendulum; for example, by touching the moving arm. The "hand" button shown in Figure 1 is used to introduce a perturbation via a software command, a feature of particular utility when the users perform remote experimentation. In the inverted pendulum case, the server simulates a perturbation by adding an error to the angle measurement. Less important information such as the sampling period or the connection states are accessible through an extra window.

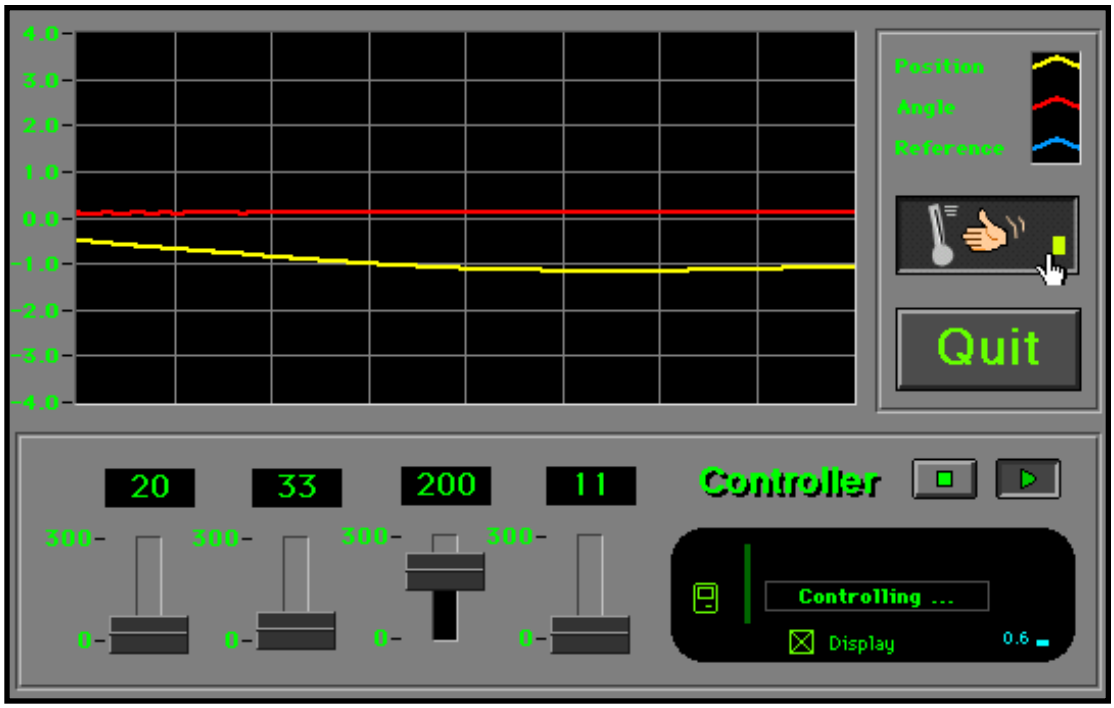

Figure 2. Graphical user interface that allows the user to interact with an inverted-pendulum experiment in real time. The area displaying a hand is used to introduce a perturbation.

3. Issues of relevance to the design

An effective remote experimentation setup must satisfy a number of requirements. In particular, the user needs to feel like he/she is physically located next to the real experiment. During local experimentation the user can use his senses of vision and hearing to perceive the effect of his acts on the control system. Under a remote experimentation mode, this specification can be addressed by providing audio and video feedback information in addition to the information given to the remote computer through the GUI. Obviously, such feedback needs to be given in a reasonable amount of time, minimizing the misleading (and most likely also annoying) effects of signal-transport delays. For example, a remote user may not accept as real-time a signal that arrives 30 seconds after an action is taken when in fact the local response is achieved in fractions of a second. Consequently, fast system responsiveness is a key goal in all developments for remote real-time control. As may be expected, ideal instantaneous responses are not possible. In our experience, 2 to 5 seconds of response time have proven to be adequate values for transatlantic experiments.

A second issue of importance is to design the overall system in a highly modular fashion. For example, our prototypes consist of three basic modules (Figure 3), namely (1) a real-time module which is responsible for executing local control actuation on the real system, (2) a GUI module that displays data and that manages the user's communication with the real system, and (3) a network module that manages all the Internet transactions. This modularity permits quick and bug-free reconfigurations of the same software to address different needs. For example, taking the GUI and the real-time modules one can build a local set-up. A network server might not need a GUI, since in this case only the real-time and the network modules may be needed. Finally, a network client only needs the GUI and the network components.

A third issue of relevance is the addition of the capability for carrying out simulation studies, where the response to all user actions are produced by a software representation of the physical process rather than by the process itself. This allows the user to evaluate different operational scenarios before attempting the real

experimentation. Note that in some complicated systems the simulation of the physical process might be difficult. Skeptics might argue that the whole idea of a laboratory experience is to work with the real process, and hence there should be no emphasis placed on simulation work. We argue later that in some cases simulation capabilities can be of significant value.

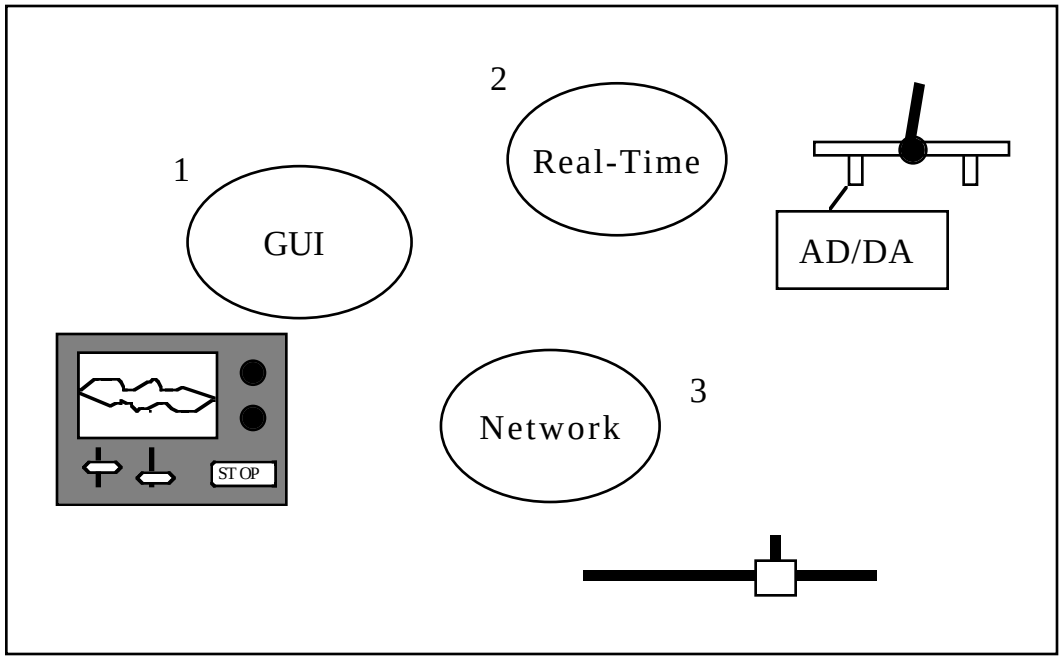

Figure 3. The three basic components of the design: (1) real-time, (2) GUI, and (3) network modules.

3.1. Brief description of the three basic modules

The three basic modules shown in Figure 3 have specific functionalities and are responsible for different security processes that ensure the robustness of the set up.

3.1.1. The real-time module

The real-time module allows the control of the real process from a computer via AD/DA boards that capture measurements signals and issue command signals. All actuations are handled by control algorithms that run on the computer's main processor. Real-time control is achieved with the assistance of a *Real-Time Kernel* [5] developed at the Swiss Federal Institute of Technology. The control algorithms are written in C or as S-functions in the format of the Matlab/Simulink software suite [6], and are executed by the real-time kernel and executed by the real-time kernel. This module must include low-level security procedures that prevent the user or the algorithms from carrying out either inadvertent or deliberate actions that might damage the physical experiment. When such procedures are activated, the real process is reset to a known safe state.

3.1.2. The GUI module

The GUI module (Figure 2) performs a display function, allowing the user to follow the time evolution of all signals of relevance to the experiment, such as the internal states of the controller, for example. In addition, through the GUI the user is allowed to also modify the parameters of the controller, as well as other adjustable characteristics of the experiment, like the sampling period for example. Higher-level of security precautions are handled at the level

of the GUI module. For example, the GUI may prevent the user from selecting physically unrealizable parameters (such as a negative sampling time), etc. The graphical user interface is based on LabVIEW. A special Real-Time Framework [5] has been developed on the Macintosh platform to ease the development of integrated real-time controllers.

3.1.2. The network module

Finally, the network module allows the program to communicate with other computers distributed in different physical locations. This module also takes care of security issues regarding network management, such as preventing unauthorized access and scheduling access in order to avoid conflicts.

4. Operation of the Remote Experimentation System

4.1. Client-server configuration

Two different kinds of software entities are involved in the communication process, namely, a server and its clients (Figure 1). The server is the local machine connected to the experiment. The server runs the algorithm used to control the experiment in real time. Although not strictly necessary, it may be convenient to use a server GUI similar to that of the clients; this is particularly useful if the server is sometimes used for local experimentation. A digital camera focused on the physical experiment and a microphone are connected to the server to respectively generate video and audio signals that are transmitted over the network.

The client software runs on the remote machines. Its main components are a network module and a GUI module. Each client can adopt two modes of operation, namely, a *standard client* or a *master client* mode, as described in the following section.

4.2. Managing client requests and assigning a master mode

The typical operation of the system is through point-to-point sessions that progress according to a hierarchy of client requests. In this approach the server is connected uninterruptedly to the physical experiment 24 hours per day, waiting for clients to request access. If there is no client connected the server might, if appropriate, stop or put the real process in an idle state to save energy.

When a user wants to perform a new experiment (*i.e.*, testing new parameters) from a remote location, he/she launches the client software and requests a connection to the server. The server allows the connection if the user has the appropriate permission and if the maximum number of connected clients is not exceeded. The user first logs in as a *standard client*, a mode that allows the observation of the experiment but does not allow issuing any commands to the physical system or its set up. In this mode the user receives audio, video, and data streams from the server.

The user can then place a request to the server for a connection as a *master client*, a mode that permits issuing commands that are accepted by the server as legitimate manipulations on the physical process. If the user has appropriate permission to become a master client, the request will be placed in a queue, and the user will remain in the standard client mode waiting for the request to be approved by the server. The server assigns the status of master client for a pre-defined period of time to the client on top of the master-client request queue. Only one client at the time can have the status of master client.

The master client may introduce manipulations to the local system, such as modifying the parameters of the controllers, and then observe their effects. At any time the user can elect to quit from the master mode and return to the standard mode, surrendering control of the experiment. The server forces the transition to the observer mode whenever the pre-assigned connect time expires.

In addition to the point-to-point mode, it is also possible to operate the system in a multi-clients session where

the master client is the instructor for the class who carries out a demonstration that can be simultaneously observed by all remote and local students participating in the class.

4.3. Classes of information streams

Different kinds of information are exchanged between the server and the clients according to four different classes, namely, (i) the parameter stream, (ii) the data stream, (iii) the administrative stream, and (iv) the audio/video stream. These streams must share the available bandwidth (which in turn might be different for each client). A comparative summary of the characteristics of all four streams is given in Table 1.

4.3.1. The parameter stream

The *parameter stream* consists of control packets sent by the master client to the server. This stream holds the highest priority. In our inverted-pendulum prototype the parameters are the four gains of the state controller, the sampling period, and the state of the "hand" button used to introduce perturbations (see Figure 2). These values need to be transmitted with a very high priority in order to assure the best possible responsiveness. This is quite easy to achieve since the amount of data transmitted is normally very small. For example for a PID (proportional-integral-derivative) controller, the data required may consist of only three values, namely the values of the P, I, and D constants. A packet is exchanged every time the user changes one of these three parameters. Since the parameter stream provides information that directly manipulates the physical system, the data transmission must be highly reliable. Consequently, additional computational requirements may be imposed by the need of encrypting and/or error control encoding the values transmitted to avoid deliberate or inadvertent corruption of the data during the transmission phase.

4.3.2. The data stream

The *data stream* consists of all signals of the physical process which are measured, such as outputs, set-points, inputs and computed internal values of the controller, for example the error between the set-points and the actual position. This stream flows exclusively from the server to the clients and has a priority just below the parameter-stream priority. All values are sent in records consisting of n measured values for each of the measured signals plus the internal values of the controller. The size of the data to be transmitted depends on the amount of data the process generates, and needs to be adapted to the available bandwidth. Solutions such as compression or decimation can be used if the amount of data produced by the process does not match the available bandwidth. The data stream is broadcast by the server to the clients. In this process, packets can be lost or may arrive out-of-order. A buffer can partially solve this problem and smooth the stream of data. The buffering time must be small in order not to increase the latency of the entire process. Another solution is to reconstruct locally the data using a real-time simulation that makes use of a model of the physical system. The model could feature the use of parameters that are constantly re-identified, updated, and broadcast by the server. In this case it is important to alert the user whenever simulated data is generated. The integrity in time of the data stream has only to be guaranteed when a data history is requested by the user (for playback or off-line analysis purposes). For live interaction, this is not quite as critical.

Table 1. Characteristics of the four types of information streams used in real-time remote experimentation.

Stream	Direction	Priority	Bandwidth	Size	Encryption	Packet Drop
parameter	client → server	highest	low	small	yes	not allowed
data	client ← server	high	high	large	no	allowed
audio/video	client ← server	medium	high	largest	no	allowed
administrative	client ↔ server	lowest	low	smallest	yes	not allowed

As an example of the size of a record in the data stream, consider a situation where the controller works with a 10 millisecond sampling period and there is a circular buffer on the server side which can hold 500 points (equivalent to 5 seconds of data). There are four values measured, and each measurement is stored in the form of short (4-byte) records. If the server sends new measured data every 2 seconds, then the packet will consists of 1600 bytes (*i.e.* 200 points x 4 values x 4 bytes).

4.3.3. The administrative stream

The *administrative stream* is mainly used at the beginning of the exchanges between a client and the server. It is also used when a standard client requests to become a master client, and vice-versa. Typical information exchanged in this stream are the *username* and the *password*; hence this stream needs to be encrypted. This data exchange goes both ways between the client to the server. The information exchanged is very small in size and has the lowest priority.

4.3.4. The audio/video stream

Finally, the *audio/video (A/V) stream* is broadcast from the server to the clients. This stream is analogous to the data stream, but it holds lower priority. The A/V stream demands the most bandwidth for timely and accurate transmission; however, in the configuration proposed it actually uses only residual bandwidth due to its lower priority. This is a key difference between the proposed paradigm for remote real-time control and the usual video conferencing/broadcasting solutions. The A/V information is important to the remote user since it provides a facility for "seeing" and "hearing" the effects of all manipulations. Clearly, better quality of the A/V transmission leads to a more satisfactory experience for the remote user of the virtual laboratory.

Some of the solutions used for the data stream can be applied to lower the bandwidth required for the A/V stream. No real distinctions are made between the audio and video part of the stream even though in our prototypes the need for the video feedback is obvious while the need for the sound feedback is more dependent to the experiment. One particular case is the example of an electrical drive set-up that produces an audible vibration if the parameters are not adequately chosen. In this case the audio signal is very important because the video image would not reveal the problem due to the small displacements of the drive that are realized. For our other prototype systems the sound does not provide useful information and can be switched off.

4.4. Hierarchy and speed requirements

The four different streams needed for remote experimentation have different transmission priorities. The amount of data for each stream varies from a few bits per second for the parameter stream to the full bandwidth for the A/V stream. Prioritization of the transmission is necessary to guarantee that the most crucial information is transmitted in a timely fashion. The parameter stream has the highest priority since it contains critical information concerning parameter modifications and other adjustments that the clients make on the physical system. A parameter modification in the client interface is sent to the server via the parameter stream and its effect is sent back to the client via the data stream. Hence, the data stream, which acknowledges all modifications and reveals their effects, receives the second-highest priority.

The cumulative round-trip time in this parameter-data exchange is the sum of the transmission times for each two streams. Typically, the time for the server to process the parameter adjustments can be neglected. Naturally the cumulative time in the exchange should be as small as possible. Our experience shows that user acceptance is very good when the cumulative time ranges from 1 to 2 seconds. When the cumulative time exceeds 5 seconds the user needs to adapt to the delay and the interactivity decreases considerably. When more than 10 seconds are needed to see the effects of a parameter modification, the user tends to resends the parameter modification because he/she did not get a visible acknowledgment. Most of the time this leads to an unwanted cycle in the loop user/remote-process, and makes the remote experimentation impracticable. In order to partially overcome this limitation, visual information in the GUI can monitor and display the state of the parameter transmission. When such a delay is present, the whole process should be switched to a batch mode where the user sends the operation to be performed to the server and watches the result in a delayed time. While this can no longer be considered real-time remote experimentation it certainly provides a workable alternative for Internet access to physical facilities. Indeed, there are strong parallels in this regard with packet telephony and packet video services where network latency is sensed periodically and the length of playback buffers is adjusted to facilitate smooth replay.

The A/V stream has the third-highest priority, since the information transmitted by this stream is to some extent redundant to the data stream. The A/V stream serves to give an added sensorial experience to the activity; hence, its role is typically less crucial. The data and the video stream should be synchronous. If the video frame rate drops the user needs to be able to figure out the relation between the video frame and the displayed measure.

4.5. Other requirements

It is desirable that the remote experimentation system should be available 24 hours a day and should require minimal local maintenance. That means the process should be able to go back to a known safe state— for example return the pendulum to the center of the track— immediately after an undesirable state or a dangerous situation is identified. Such undesirable situations may occur due to a number of reasons, including a user action, the selection of wrong parameters, a network problem or a loss of connection, etc. Precautions should be taken so that the physical system is protected from damage in all contingencies.

On the other hand, the precautionary procedures should allow the user to operate sufficiently close to undesirable states so that he/she can learn from the experience. For example, specifying a large gain in the controller that adjusts the arm position will cause the arm to develop position oscillations. This will progressively move the arm to one end of the track where it will activate a built-in position sensor that disconnects the controller and safely reposition the pendulum away from the end of the track. Of course, the precautionary procedures should be robust enough to guarantee recovery from unwanted states, such as when the user specifies exceedingly large controller gains.

The user in a remote location should also be able to perturb the physical process in order to evaluate the ability of a set of control parameters to reject the effect of the induced perturbation. For example, in the case of a mechanical system, a brake could be used to change the acceleration of the process. A mechanism for performing

such perturbations must be integrated in the software and made available to the remote user. In the case of our inverted pendulum prototype, when the user selects a perturbation box featuring the drawing of a hand (see Figure 2), an arbitrary bias is temporarily added to the signal sent to the controller, hence effectively inducing a large step-change type of disturbance on the arm position.

5. Remote Experimentation vs. Videoconferencing/Broadcasting

A growing collection of software is now available to transmit or broadcast audio and video through the Internet. One may argue that video broadcasting/conferencing is similar to the remote experimentation paradigm, and indeed, this analogy does hold in the sense that both send audio and video images. Furthermore, the data stream (measurements) can conceivably be interpreted as audio or video data. However, there are significant differences.

The main difference between video conferencing and real-time remote experimentation lies in the fact that in the latter the emphasis is on transmitting and receiving the most recent data, even if this is done at the expense of discarding older or out-of-order data. In contrast, video conferencing software tries to send all the data, recent and older, without any perceivable losses from the user's point of view. Consequently, if possible, no information is dropped, and buffer sizes are increased in order to smooth the transmission. Note that under this perspective buffers should be avoided as much as possible in real-time remote experimentation.

The original version of our system used a commercial off-the-shelf software [6] to transmit audio and video signals. This solution gave acceptable delays— approximately one to four seconds— for experimentation conducted within the local university LAN. Unfortunately the performance became unacceptable for transatlantic experiments because the delays became significantly greater and were highly unpredictable. Since in this case the stream was not adaptively managed by the server, it was not possible to synchronize the A/V stream with the data stream and manage their relative priorities. This problem is more important than transmission delays because the user may be misled about the actual state of the process and may not be able to determine which information to trust.

Another difference lies in the fact that in video-conferencing/broadcasting preference is given to the audio over the video since the human ear is very sensitive to the continuity of the sound. Interruptions of the sound stream, even for short periods of time, are disturbing for the user. In contrast, the opposite is true for the vision since the eye can easily adapt to "jerky" images. However, with some exceptions, in remote experimentation the video stream is often more important than the audio stream because the visual information plays a larger role in making the user participates in the experiment with sensory information.

Most video conferencing/broadcasting software tend to increase the buffer at the reception end in order to smooth the transmission. Increasing buffers means adding delays which does the opposite of what the remote experimentation tries to do. In the later case older (out of order) information is dropped if newer information is available. However, the removal of buffers has its drawbacks. For example, buffers are needed for compression, especially for video signals where more than one video frame is required by high compression-rate algorithms. Video compression may be in some cases unavoidable, depending on the size and the number of colors of the image to be transmitted. Typical compression schemes do not send any data if the image does not change between two consecutive frames. Naturally, when the image changes rapidly the compression scheme delivers a large amount of data. By lowering the frame rate one can reduce the size of the transmitted data, but this introduces some drawbacks. Lets take the example of a frame rate originally set at 10 frames per second which is to be reduced to 1 frame per second. If the codec (coder-decoder) needs the 3 last images to perform the compression, it will take about 1/3rd of a second in the former case and 3 seconds in the latter case. Hence, the compression time increases proportionally to the compression ratio.

6. Optimizing the Use of the Available Bandwidth

In order to use the available bandwidth efficiently the transmission rate of the different streams needs to be

adjusted based on the respective stream priorities. Different techniques can be used to lower the required data rate to make better use of the available bandwidth. The first technique is data compression, but it involves a trade-off because of the additional delay introduced through the compression and decompression operations. This delay should be kept much smaller than the transmission delay.

Since the A/V stream requires the most bandwidth great attention should be taken to manage it carefully. Video conferencing/broadcasting offers a compression rate of up to 50:1; however, such a ratio can only be achieved when the compression is done in both the space and time dimensions. These solutions expect a stream as continuous as possible, which might not be the case in remote experimentation.

Another problem arises when clients are not on the same network. For example, some might be on the local LAN while others may be connected from home using dial-up line. The server needs to adapt to these different bandwidth requirements. One solution for the video stream is for the server to layer the image and send it at different resolutions [7]. When a low bandwidth is available the client would only use a low-resolution image. When more bandwidth is available the client would use the low-resolution image and also the higher layer image.

Data decimation, where only one sample over *n* samples is transmitted, can also be used to lower the bandwidth required. The intermediate steps can be reconstructed on the client's end where interpolation or a real-time simulation can be used to regenerate the missing measurements. However, if two measures are too distant in time the reconstruction can miss fast phenomenon such as high-frequency oscillations.

A virtual-reality model can also be used as a replacement of the video signal. In this case the client software animates a graphical representation of the physical process using regularly updated coordinates provided by the server via the data stream. Another concept called phantom process [8] uses the local simulation to directly reflect the effect of the user actions (*i.e.*, parameter modification) without waiting for the server to send the information back. At the client's end the user sees two windows: one free of delay but showing the results of the phantom-process simulation, and one with a transmission delay but showing the image of the physical process. This technique is used in robotics applications where delays due to distance cannot be avoided. Phantom-process schemes often require the use of some kind of mechanism, such as artificial intelligence schemes, to deal with unmeasured and/or unknown events that may be missed by the base model.

7. Conclusions

Remote-control experiments over the Internet are feasible for long-distance applications, such as transatlantic communications, via a paradigm that involves a client-server structure. The requirements of these systems are significantly different from those of conventional broadcasting or videoconferencing systems particularly due to the enhanced emphasis given to the successful transmission of recent data to the detriment of older data. Experience shows that multiple aspects must be taken into consideration to obtain adequate performance, including a system for prioritizing information streams and the utilization of appropriate data compression mechanisms. The interested reader may test real-time remote experimentation by accessing the Telepresence Web Server (http://iawww.epfl.ch) at the Swiss Federal Institute of Technology.

Acknowledgments

The first and third authors gratefully acknowledge support received from the Fonds National Suisse under grant number SPP-ICS 5003-045347. The fourth author gratefully acknowledges support received from the National Science Foundation under grant number CTS 9502936.

References

[1] Gillet D., R. Longchamp, and D. Bonvin, “Integrated Workbench for Laboratory Projects in Automatic Control.” *Int. Conf. on Computer Aided Learning and Instruction in Science and Engineering*, Lausanne, Switzerland, September 1991.

[2] Gillet D., C. Salzmann, R. Longchamp, and D. Bonvin, "Telepresence: an Opportunity to Develop Real-World Experimentation in Education". *European Control Conference*, Brussels, July 1997.

[3] Access the electrical-drive experiment at http://iawww2.epfl.ch/drive and the Helicopter experiment at http://iawww2.epfl.ch/toycopter.

[4] National Instruments, Austin, TX.

[5] Salzmann C., D. Gillet, R. Longchamp, and D. Bonvin, “Framework for Fast Real-Time Applications in Automatic Control Education”. *IFAC Symposium on Advances in Control Education*, Istanbul, July 1997.

[6] *Connectix Videophone*, Connectix Corporation, San Mateo, CA

[7] The MathWorks Inc., Natick, MA.

[8] Vetterli M., Jacobson V., and S. McCanne, “Low-Complexity Coding for Receiver-Driven Layered Multicast.” http://sscwww.epfl.ch/Pages/publications/ps_files/tr97_001.ps *Technical Report SSC/1997/001*.

[9] Kevin J. Brady, *Timed-Delayed Control of Telerobotic Manipulators*. Ph.D. Thesis, Washington University, August 1997.