Integration of real-time hardware and TCP/IP communications for teleoperation systems

Ollin Peñaloza-Mejía*, Luis A. Márquez-Martínez** and Jaime Álvarez-Gallegos*

***Abstract*— This paper describes the development of an experimental platform for robotic teleoperation via Internet. It is obtained by adding Internet capabilities to an industry-standard DSP board, available in many universities and research centers.**

***Index Terms*— Teleoperation, Internet, TCP/IP, real-time hardware.**

I. Introduction

The use of Internet as a communication medium can provide cost-effective, flexible and easy-to-access control systems that are not limited to any geographical region. As a consequence, it is natural to think in applications of telecontrolled systems operating in different environments. However, long distance control of systems (independently of the transmission media being used) causes two main technical problems: limited bandwidth and transmission delays due to the propagation. These constraints result on one hand in difficulties for the operator to securely control the remote system and, on the other hand, that the use of classical control schemes may result in poor performance or even instability.

Although the implementation of real-time control algorithms over the Internet involves random time delay in the control loop, many approaches have been proposed in recent years to either eliminate or to mitigate its detrimental effect on the stability of the system. The foremost approach to manage the varying network-induced delay is the introduction of buffers and to consider a greater constant time delay [1].

The problem of controlling a real-time telerobotic system using the Internet as the link has been extensively studied over the past few years. Most researches have tended to use TCP/IP protocol (with its inherent shortcomings in the ability of deliver data in a timely fashion), seemingly without considering other Internet protocols available [2]. This could be owing to TCP's features of reliable data transfer and its automatic retransmission mechanism for recovering lost packets and quality improving. Besides, it has been shown that a slight change to TCP can make it suitable for real-time applications [3].

The aim of our project is building and configuring a reliable setup for bilateral teleoperation over the Internet, in which new control algorithms will be implemented and evaluated. It will also be used for implementing virtual laboratories in engineering education to train students in performing experiments for the classical control courses. It is based on an industry-standard controller board, with the advantage of its very intuitive and easy-to-use graphic user interface. In addition, the same approach can be used for other boards from the same company.

The overall setup consists of two workstations connected to a communication channel (Internet) which induces a time delay in the information transport (Fig. 1). Each of the stations has an electromechanical system which is locally controlled by real-time hardware. The one used by a human operator to generate the desired velocity, position and force variables is named the master mechanism while the one that uses this information to perform a task in a remote environment is named the slave mechanism.

A methodology to integrate real-time hardware and TCP/IP communications for applications of remote robotic manipulation is presented and validated in this work. Such integration is achieved within the MATLAB [4] environment by means of implementing special functions that can send and receive data via a TCP/IP connection while accessing real-time hardware to control the teleoperation system.

The rest of the paper is organized as follows. Section II describes the methods and materials considered to make the integration. Section III presents the selection of certain libraries and the way they are integrated. Section IV shows experimental results of the teleoperation platform, and finally in section V concluding remarks are given.

II. Materials and Methods

A. Real-time hardware

A platform for Rapid Control Prototyping with Internet capabilities is required in the project. For this a digital signal processor (DSP) controller board from dSPACE (DS1104) [5] has been used. The reason of selecting this hardware is because of its easy-to-use features and rapid design of control algorithms from SIMULINK. Unfortunately, this board does not supply SIMULINK blocks to establish an Internet connection within a control loop, nor accepts the ones available for SIMULINK.

B. Sockets

One problem was the lack of Internet capabilities of the selected DSP board. For this the solution required: (1) the implementation of a TCP/IP socket from the MATLAB workspace which can communicate both workstations and (2) the use of the exchanging data to access and modify

* CINVESTAV-IPN, Depto. de Ingeniería Eléctrica, Sección de Mecatrónica. Ap. Postal 14-740, 07600 Mexico, D.F., MEXICO. {openaloz, jalvarez}@cinvestav.mx

** CICESE, Depto. de Electrónica y Telecomunicaciones. Km 107 Carr. TIJ-ENS, 22860 Ensenada, B.C., MEXICO. lmarquez@cicese.mx

0-7803-9419-4/05/$20.00 ©2005 IEEE

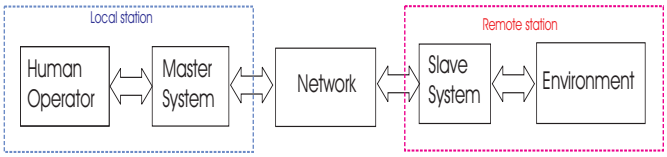

Fig. 1. Block diagram of the overall setup for teleoperation.

control parameters or variables in the real-time processor (which is currently controlling the mechanical device).

As it was shown in [6], this can be achieved by means of using a dSPACE library called CLIB with a Common Gateway Interface program. In this work, an alternative methodology by using the dSPACE MLIB/MTRACE interface libraries is explored.

The dSPACE interface libraries give access to the real-time processor hardware from the MATLAB workspace. This is done by the MLIB/MTRACE functions (implemented as MEX DLL files) which provide real-time data capture capabilities making them suited to modify parameters online, generating interrupts, setting the processor state and getting processor status information.

The way the workstations exchange data is based on a client/server architecture and it is performed by opening an Internet connection. The following options based on free software to add TCP and/or UDP sockets from the MATLAB workspace were considered:

- DODS [7]. DODS/OPenDAP is a software framework that allows simple access to remote data through a web server via URL. This software has an utility named DODS MATLAB GUI for accessing and transporting data directly from the MATLAB workspace.
- TCP/IP SIMULINK Blocks [8]. These server and client blocks can be added to any SIMULINK model to exchange data between computers. These blocks are implemented using C MEX S-functions and Winsock2.
- TCP/IP/UDP Toolbox [9]. This toolbox can be used to set up TCP/IP connections or to send/receive UDP/IP packets in MATLAB. It can transmit data over the Intranet/Internet between MATLAB processes or other applications. It is possible to act as server and/or client and to transmit textstrings, arrays of any datatype, files or MATLAB variables.
- IOLIB [10]. This library allows port and memory IO for MATLAB and SIMULINK. It also has several functions to install and execute different commands such as timers, interrupts and TCP/IP communications to perform soft real-time applications.

C. Software

The software utilized for implementing the teleoperation setup is the following:

- SIMULINK 4 and 5: For graphically programming the control algorithms.
- MATLAB 6.1.0.45(R12.1) and 6.5.0.180(R13): For implementing the Internet connection (by installing a TCP/IP socket) and accessing the real-time processor.

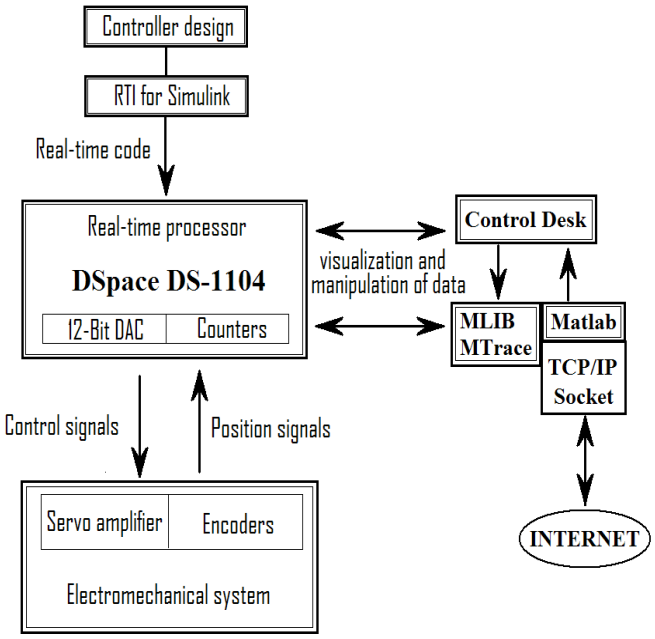

Fig. 2. Architecture of the proposed solution for each workstation.

- RTI 4.3: For generating and installing the real-time code in the DSP.
- ControlDesk 2.2: For displaying real-time information.

III. Selection and Integration

After reviewing and testing the foregoing tools for implementing sockets from the MATLAB workspace, it has been decided to use in the integration the IOLIB set of functions. This is because the other tools have some restrictions for implementation purposes, such as non interactive data exchange in a real-time fashion (DODS and TCP/IP/UDP Toolbox) or because they cannot be installed on the real-time processor (TCP/IP SIMULINK Blocks) for limitations to a simulation environment.

Exchange of data online between both workstations is done through the network connection by installing a TCP/IP socket within the MATLAB environment. The key is to deploy the TCP/IP communication and timer tools from the IOLIB set of functions (*tcp.m* and *itimer.m*) together with the MLIB/MTRACE functions in a pair of M-files. By doing this, communication and control for both stations in a soft real-time fashion is achieved. The architecture for each workstation is shown in Fig. 2.

The controller for the mechanism is designed from SIMULINK considering the information to be received and transmitted as data variables. Then the RTI generates the real-time code which is uploaded to the DSP. The MTRACE library allows these variables to be shared by the MATLAB workspace and the real-time processor. Therefore, the complete integration is achieved within the MATLAB workspace by installing the TCP/IP socket and using the MLIB/MTRACE libraries to identify and modify these variables online.

The reader should refer to the appendix for an overview of the use of the IOLIB and MLIB/MTRACE functions to update (read and write) the variables in the real-time processor at the same time that M-variables are sent to the remote workstation (client) and N-variables are received at the local workstation (server).

It is important to notice that IOLIB functions work using multithreading. However, MATLAB's command windows and graphical user interfaces are not intended for this operation mode, which results in a software crash. To avoid this, MATLAB must be started without the Java Virtual Machine (using the -nojvm option when invoking it from the command line).

IV. Experimental results

A simple teleoperation system consisting of a couple of identical 1-degree-of-freedom nonlinear mechanisms (pendulums) has been regarded to show the experimental performance of the integrated setup in a local network. At the local station (master side configured as the server) a physical pendulum is controlled by the real-time hardware from dSPACE (DS-1104 board) and at the remote station (slave side configured as the client) a virtual pendulum is simulated in real-time (Fig. 3).

A. Plant

The plant considered is a mechanical pendulum (manufactured by Mechatronics Systems Inc), whose motion is governed by the equation

$$\alpha_1 \ddot{x}(t) + \alpha_2 \sin(x(t)) = \tau(t), \tag{1}$$

with physical parameters $\alpha_1 = 0.0143$ K$_g$m^2 and $\alpha_2 = 0.9976$ Nm.

Considering for the master system the new variables

$$x_1^m = x, \quad x_2^m = \dot{x}, \tau^m = \tau, \tag{2}$$

the following state equations are obtained

$$\dot{x}_1^m(t) = x_2^m(t) \tag{3}$$

$$\dot{x}_2^m(t) = \frac{1}{\alpha_1}[\tau^m(t) - \alpha_2 \sin(x_1^m)]. \tag{4}$$

The state equations for the slave system are defined in a similar vein using the variables $x_1^s(t)$, $x_2^s(t)$ and $\tau^s(t)$.

Defining

$$\tau^m(t) = \alpha_2 \sin(x_1^m) + \alpha_1[u^m(t) + u^h(t)] \tag{5}$$

$$\tau^s(t) = \alpha_2 \sin(x_1^s) + \alpha_1[u^s(t) + u^e(t)], \tag{6}$$

then the simplified master/slave model is obtained

$$\dot{x}_1^m(t) = x_2^m(t) \tag{7}$$

$$\dot{x}_2^m(t) = u^h(t) + u^m(t) \tag{8}$$

$$\dot{x}_1^s(t) = x_2^s(t) \tag{9}$$

$$\dot{x}_2^s(t) = u^e(t) + u^s(t), \tag{10}$$

where $u^m(t)$ and $u^s(t)$ are the new control inputs while $u^h(t)$ and $u^e(t)$ are the inputs caused by the human and the environmental force, respectively.

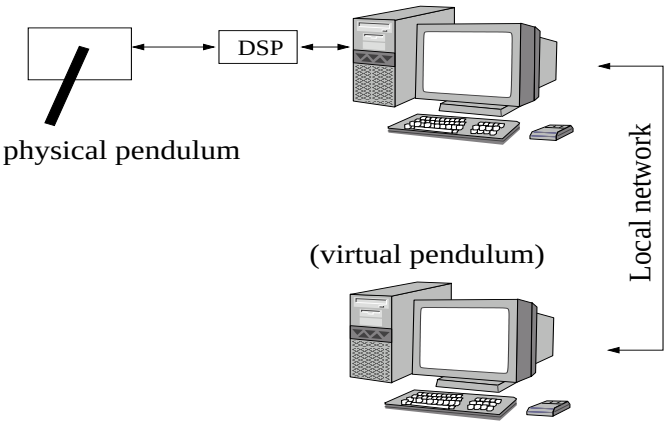

Fig. 3. The teleoperation setup used for experimentation purposes.

B. Control scheme

Although all the control schemes reported in [12] were tested in the teleoperation platform proposed in this document, due to space reasons only one of them is presented: the so-called Force Reflection (FR) in which position information is transmitted from master to slave and force information flows in the opposite direction. The control equations are [12]

$$u^m(t) = u^s(t - \Delta), \tag{11}$$

$$u^s(t) = -k_p[x_1^s(t) - x_1^m(t - \Delta)], \tag{12}$$

where k_p is the control gain and Δ is the network time-delay, which under our approach is considered to be constant, assumption that is justified later in this document.

C. Implementation

Several experiments for teleoperation were conducted within an Intranet environment (100Mbs Ethernet network) for the workstations depicted in Fig. 3. The local station is mainly composed by the physical pendulum controlled by the dSPACE board DS1104 in a Pentium IV PC with Win2000 Pro and MATLAB 6.1(R12.1), while at the remote station the dynamics of a virtual pendulum is simulated (using a standard fixed-step 4th-order Runge-Kutta method [11]) in a Pentium IV Laptop Toshiba with WinXP Pro and MATLAB 6.5(R13). The control gain k_p was set to 50.

The maximum period for sending or receiving data that can be achieved in the platform is 1ms. This is because within the tested local network, the TCP/IP protocol needs a mean time of 0.458ms to transport one packet from one station to the other. However, when transporting information over the Internet for long distances, this period must be greater than the time needed for the network to transport packets in one direction.

As it has been shown in [1], the implementation of buffers in the data transmission allows to use a greater constant delay in the network for developing teleoperation experiments over the Internet even when a lost of packages could exist. Obviously, this constant delay could be used for setting the period of data exchange. Another way for setting this period is by using a Round Trip Time (RTT) tool.

In the experiments presented in this work, it has been considered the case when partial information at each exchange period is available, so buffers are not used. The sampling period was set to 1ms at each workstation and they were programmed to exchange data every 40ms (greater than the RTT needed to transport a packet over the Internet from Ensenada to Mexico City). Notice that the delay in the network does not matter as long as the information (transmitted or received) is available at each period of data exchange. This delay in the network could be varying but as the information is being updated periodically, the delay could be regarded as constant.

After initialization, the server opens the Internet connection and waits for the remote station to connect. If the connection is established, the control runs at both stations while accessing the real-time hardware and exchanging data online in accordance with the desired trajectories that the human operator is currently generating. Eventually, if an external force acts at the slave side, it is reflected to the operator resulting in an increase of the telepresence sensation. Below are summarized the steps needed to perform the experiment.

Server

- Program the control scheme (11) from SIMULINK.
- Generate the corresponding real-time code and upload it to the DSP board by using the RTI.
- Run MATLAB without the Java machine.
- Initialize the server by running from the MATLAB workspace its M-File (see the appendix). At this point, the server has already identified the variables to be exchanged and start to listen to the remote connection.
- Once the client is connected, the human operator manipulates the pendulum.
- The Internet connection is closed once the experiment is finished.

Client

- Run MATLAB without the Java machine.
- Program the control scheme (12) and the dynamics of the pendulum in a M-File.
- Connect the client to the server by running from the MATLAB workspace its M-File (see the appendix). At this point, the client has already identified the variables to be exchanged and waits for the first packet to begin the simulation.
- The client disconnects once the experiment is finished.

D. Results

For the FR control scheme, Fig. 4 shows the positions and torques of the pendulums. It can be seen that the slave tracks accurately the master just before that an external constant force (also simulated) acts at the slave (Fig. 5, lower diagram) at $t = 15$, causing that position drift appears. It is also shown that during the experiment, apparent inertia appeared. As it is well known in teleoperation, these features are typical of the FR control scheme.

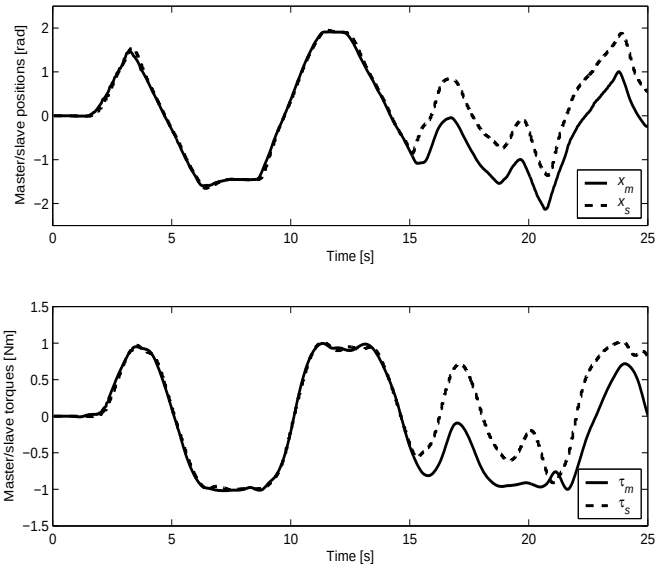

Fig. 4. Positions and torques for the master and slave pendulums.

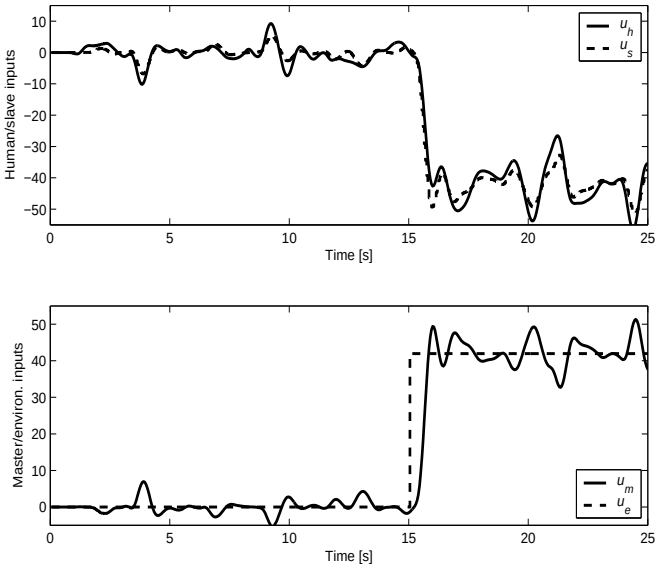

Fig. 5. Control inputs for the master and slave pendulums.

Notice that u^s tracks u^h while u^m tracks u^e (Fig. 5, upper diagram) showing that some typical control objectives for this kind of teleoperation systems are being satisfied as much in free movement as in constrained movement.

V. Conclusions

In this work, Internet capabilities has been added to the industry-standard DSP board DS1104 from dSPACE to develop a teleoperation Rapid Control Prototyping platform, with the easy-to-use graphical user interface included with such board. This platform can be used for testing teleoperation schemes over networks (Internet/Intranet) and for distance teaching.

VI. Acknowledgments

This work was partially supported by CONACYT Mexico (first author's scholarship 175135) and the French-Mexican Laboratory on Automatic Control (LAFMAA) through the project PRODIGE.

References

[1] P. Berestesky, N. Chopra and M. W. Spong, Theory and experiments in bilateral teleoperation over the Internet, *in Proceedings of the IEEE International Conference on Control Applications*, Taiwan, 2004.

[2] L. Bate and C. Cook, The feasibility of force control over the Internet, *in Proceedings of the Australian Conference on Robotics and Automation*, 2001, Sydney, pp. 146-161.

[3] S. Liang and D. Cheriton, TCP-RTM: Using TCP for real time applications, *in Proceedings of the IEEE International Conference on Network Protocols*, 2002.

[4] MATLAB and SIMULINK for technical Computing, web site: `http://www.mathworks.com`

[5] dSPACE, Solutions for Control, web site: `http://www.dspaceinc.com`

[6] A. Strivastava and W. Kim, Internet-based supervisory control and stability for time delay, *in Proceedings of the American Control Conference*, Denver, CO, 2003, pp. 627-632.

[7] DODS/OPenDAP, *Distributed Oceanographic Data System and Open-source project for a network data access protocol*, available at `http://www.unidata.ucar.edu/packages/dods/`, 2004.

[8] C. Jadhav, *TCP/IP Blocks for Simulink*, available at the MATLAB Central `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4934&objectType=FILE`, 2004.

[9] P. Rydesäter, *TCP/UDP/IP Toolbox 2.0.5*, available at the MATLAB Central `http://www.mathworks.com/matlabcentral/file-exchange/loadFile.do?objectId=345&objectType=file`, 2001.

[10] W. Zimmermann, *IOLib library for Matlab*, available at `http://it.fht-esslingen.de/ zimmerma/software/IOlib.htm`, 2003.

[11] R. Beckett, *Numerical calculations and algorithms*, Mc Graw Hill, New York, NY; 1997.

[12] P. Arcara and C. Melchiori, Control schemes for teleoperation with time delay: a comparative study, *Journal of Robotics and Autonomous Systems*, vol. 38, 2002, pp. 49-64.

Appendix

In this section, the use of the IOLIB and MLIB/MTRACE functions in a pair of M-files to exchange data online is presented.

A. Server

%========== Listing of client.m ==========
% Array where the received N-variables will be stored
r=zeros(1,N);
% Selection of the dSPACE board
mlib('SelectBoard','DS1104');
%Identify the shared variables (N+M) in the DSP
variables = {'Model Root/var$_1$/Value';,...
$\vdots$
'Model Root/var$_{N+M}$/Value'};
[var$_1$ ... var$_{N+M}$] = mlib('GetTrcVar',variables);
% Auxiliar variables to read and write
escribe='write'; lee='read'; datos='data';
% Write the received N-variables down to the DSP
receive_var=mlib(escribe,var$_1$,datos,r(1));,...
$\vdots$
mlib(escribe,var$_N$,datos,r(N));
% Read from the DSP the M-variables to send
send_var=[mlib(lee,var$_{N+1}$) ... mlib(lee,var$_{N+M}$)];
% Open the Internet connection, receive and send data
tcp(1,' ',port,'r','receive_var; tcp(2,[1 ... M],send_var);');
% Wait to finish the experiment
disp('Press any key to stop the TCP/IP server'); pause;
% Close the Internet connection
tcp(0);

B. Client

%========== Listing of client.m ==========
% Time and exchange period of data
t=0; h=0.040;
% Array where the received M-variables will be stored
r=zeros(1,M);
% Selection of the dSPACE board
mlib('SelectBoard','DS1104');
%Identify the shared variables (N+M) in the DSP
variables = {'Model Root/var$_1$/Value';,...
$\vdots$
'Model Root/var$_{N+M}$/Value'};
[var$_1$ ... var$_{N+M}$] = mlib('GetTrcVar',variables);
% Auxiliar variables to read and write
escribe='write'; lee='read'; datos='data';
% Write the received M-variables down to the DSP
receive_var=mlib(escribe,var$_{N+1}$,datos,r(1));,...
$\vdots$
mlib(escribe,var$_{N+M}$,datos,r(M));
% Read from the DSP the N-variables to send
send_var=[mlib(lee,var$_1$) ... mlib(lee,var$_N$)];
% Connect to the server and receive M-variables
tcp(1,'server_IP_address',port, 'r','receive_var ;');
% Install timer to exchange variables
itimer(1,1,round(h/0.001),'add_client');

%========== Listing of add_client.m ==========
% If time exceeds the experiment duration, close connec-
% tion; else, send N-variables through N-channels over
% the network. Increase time.
if t>=ts
    tcp(2,[1 ... N], zeros(1,N));
    itimer(0); tcp(0);
else
    tcp(2,[1 ... N],send_var);
end
t=t+h;