AC 2011-1070: A UNIFIED FRAMEWORK FOR REMOTE LABORATORY EXPERIMENTS

Claudio Olmi, University of Houston

Claudio Olmi is currently pursuing a Ph.D. in Electrical Engineering at the University of Houston. He received his B.S. and M.S. degree in Computer and Systems Engineering from the University of Houston. He specializes in System Integration of hybrid Mechanical and Electrical systems with focus on Software Programming, Analog and Digital Hardware Design, Internet Technologies for Remote Operations, Digital Controls, and NI LabVIEW Programming. Olmi worked in projects using Smart Materials applied to Civil and Mechanical Structures for in laboratory and remote operations from where he published 2 journal papers and several conference presentations. He is a student member of IEEE.

Bo Cao, Smart Materials and Structures Laboratory
Han Wang, University of Houston

Han Wang is currently a PhD student of Mechanical Engineering in University of Houston. His research interests are Intelligent Controls, Nonlinear Control Systems and Modeling, Fault Detection and Isolation, and Control of Smart Materials.

Xuemin Chen, Texas Southern University
Gangbing Song, University of Houston

A Unified Framework for Remote Laboratory Experiments

Abstract

Developing a remote experiment requires knowledge in hardware and software. Nowadays, most of the great engineering experiments are not transformed into an online experiment due to the lack of complete and easy to use software solution. In this paper, the progress of developing a unified framework for next generation remote experiment laboratory is presented. The framework allows using the latest Internet technology, Web 2.0, to provide an interactive user interface for the client computer that does not need any additional software to be installed. Moreover, the dynamic interface is compatible with most of the web browser software and most of the operating systems currently available. The use of a pure JavaScript environment for the client interface provides a wide compatibility with current technologies. On the other hand, a proxy server was added between the experiment and the Internet client to hide the remote experiments from the Internet. The proxy filter route the data exchanged with the available experiments. As a result, the number of experiments behind the server could grow exponentially without changing the way or the Internet address of the collection of experiments.

Introduction

Laboratories have always been an essential part of engineering education. Even throughout secondary education, concepts that are taught through lectures have been reinforced by hands-on experience in laboratory experimentations. These experiments allow students to experience the scientific method by following the classical model of using their own experience, forming a conjecture, deducing a prediction from that hypothesis, and affirming the consequent[1,3,7].

Conventional engineering laboratories allow students to perform learning activities using their visual and tactile sensory system compared to a virtual laboratory, where only the visual perception is trained[5]. Unfortunately, the cost of setting up and maintaining a physical laboratory is becoming an obstacle for school administrations. A mid point solution to the aforementioned issue is the use of remote laboratories that allow students to perform real experiments over the Internet[9,2].

Remote laboratory experiments are defined as real experiments conducted by Internet users. These experiments use real instrumentation and components at a different location than where they are being controlled by the user. One of the popularly deployed technologies today for remote panel over the Internet is National Instrument's Laboratory Virtual Instrument Engineering Workbench (LabVIEW) software[4,6,8]. Despite the effectiveness and rapid prototyping of the software system, it has been plagued by software issues from LabVIEW's runtime engine. The runtime engine that requires administrator privileges cannot be installed on any public computer by normal users force students to use their own computer at home and therefore limits the time availability to the students. Furthermore, any updates to the LabVIEW front panel can result in version errors in the client due to the non-compatibility of the runtime engines. Recently, LabVIEW integrated a new feature to interact with the experiment Virtual Instruments by using RESTful web services. REST (Representational State Transfer) provides a

lightweight protocol accessible to a wide variety of clients. The architecture does not require complex message passing and provides a simple interface for user to begin using Web services in LabVIEW. However, it requires the client interface to be developed using different technologies. In addition, as the number of remote experiments increases, the software is not capable of handling multiple users with multiple resources.

Methodology

The primary goal of the developed unified framework is to allow the set up of a distributed network of online experiments that works in any Internet browser without the need of any extra plug-in. The project includes three sections: client side, web server, and experiment server. Figure 1 shows the simplest remote laboratory with a single online experiment. Multiple experiments would connect to the same local web server. The client side browser loads a Web 2.0 interface from the web server that is specifically designed for the running experiment. When the client is ready to run the experiment, the user starts the connection to the web server by requesting the use of the experiment. Afterward, the web server sends the "Start" command to the experiment server. The latter sends back a packet containing the last data array generated by the experiment internal sensors, actuators, and controls. The received data array is put into a database containing all the live data corresponding to the specific experiment. Finally, the client periodically requests the live data from the web server. The database acts as a communication buffer between experiment and client.

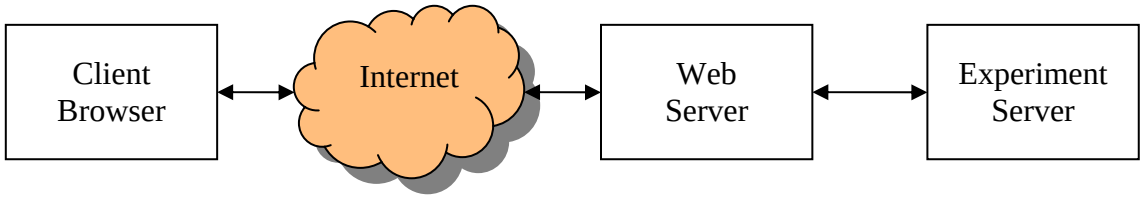

Figure 1: Simplest remote laboratory network

To allow the client browser to update the web page without refreshing the whole content, a Web 2.0 strategy was used. The unified framework was built by multiple component levels from the client to the experiment server.

Table 1: Framework levels and their corresponding technologies

Level	Name	Technology/Protocol
1	Client – Web Application	JavaScript, XHR
2	Data Protocol – Low Speed	TCP Sockets
3	Server - Web Service	WSGI, REST
4	Data Protocol – High Speed	TCP Sockets
5	Experiment Server	LabVIEW

Table 1 above shows the technology/protocol that is currently being used in the developed framework. The first level provides the user interface, or Web Application, to the client

computer using a standard web browser running JavaScript functions to pull and display information from the server. To test all the functions and the initial data protocol specification between client and server, the software was initially written in .NET C# for rapid prototyping and simplified debugging of the interface. This front end was paired with the server side software also written in .NET C# capable of retrieving and saving information on a local MySQL database. After the data protocol implementation reached a stable point, the client side software was ported to JavaScript, while the server side was being rewritten in Python scripting language. Although the data protocol scheme has not being finalized, it uses three main technologies: XHR, WSGI and REST.

XML HTTP Request (XHR) are a set of Application Programming Interface (API) functions, available in most web browsers, capable of initiating an HTML request from the JavaScript running on the client computer, and update the Document Object Model (DOM) in real time without the need of reloading the web page. Thus, it provides a method to continuously updating tables and/or plots on the web page dynamically. This technology is used at level 1. Moving to the server side at level 3, Web Services Gateway Interface (WSGI) is an interface between the web server core software and the web application that allows building web services faster then the old Common Gateway Interface (CGI) using Python language. WSGI does not provide a protocol but just a link to the underlying core software for speed.

Lastly, Representational State Transfer (REST) provides a methodology for handling packets of data from the web application. This technology is at the base of the data protocol. REST uses standard HTTP methods, GET, POST, PUT, and DELETE, to interact with the web application. Each method, called verb in REST, generates a well understood action on the server. As an example, when the web application sends a POST method with embedded data, the web service will interpret the method as a request to UPDATE data.

At the fifth level, the experiment communicates with the web server through LabVIEW. LabVIEW will acquire data from a local experiment and parse this data into an array. This array will be saved and sent to the web server through a TCP socket, which will asynchronously relay this information to the user. The user interface will then interpret the data and update part of the webpage accordingly to display the data.

Results

In this paper, the LabVIEW-based Remote Nano Fiber Beam experiment was modified to connect to the newly developed framework. The initialization of the experiment server consists of a TCP socket listening for the web server “Start” command. When the command arrives, the experiment server starts the main program loop that consists of experiment initialization, control, and collection of ten data points acquired at a frequency of 100 Hz. Every ten loops, the data is sent back to the web server. All the parameters can be adjusted by the experiment developer. The experiment will run continuously until a “Stop” command is received. These procedures constitute a complete data transmission cycle. The connection between the experiment server and the web server closes when the experiment is finished. Figure 2 shows the flowchart of the aforementioned cycle with the LabVIEW block diagram.

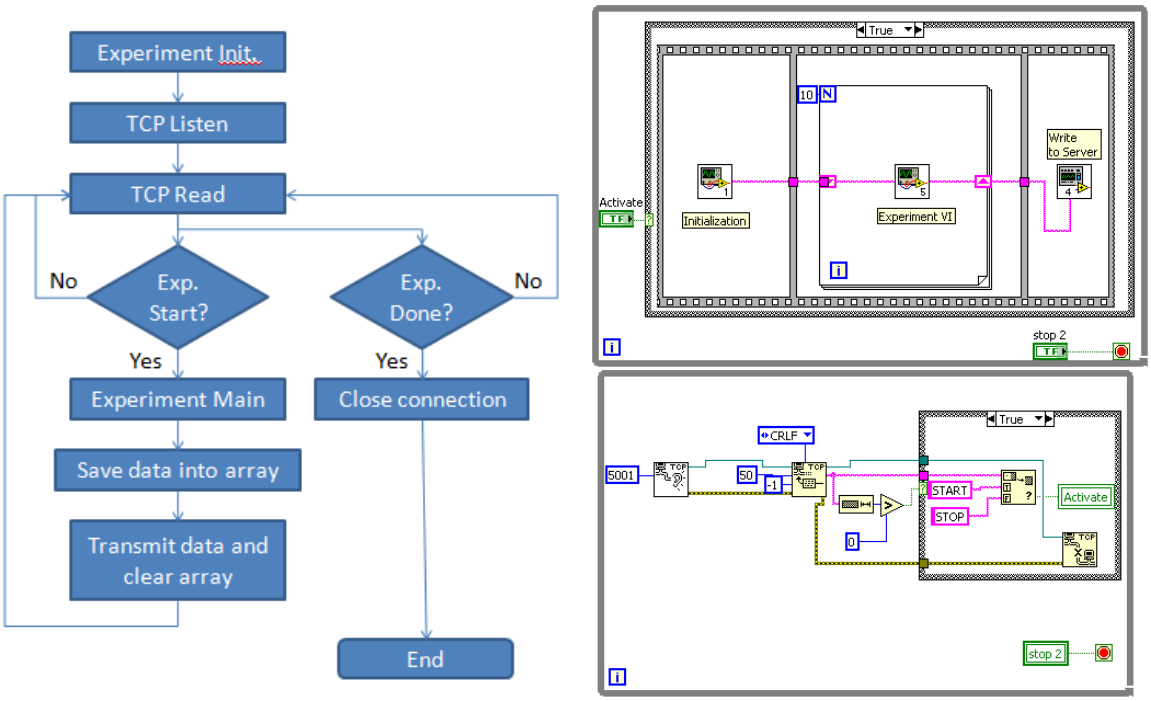

Figure 2: Flowchart logic of the LabVIEW program with the LabVIEW block diagram.

On the client side, the interactive media-rich interface is first loaded from the remote laboratory web site located in the web server. The client starts up the web page using any modern web browser such as Opera, Apple Safari, Mozilla Firefox, Internet Explorer, and Google Chrome. As long as JavaScript is enabled in the browser, no plug-ins or add-ons will be necessary to run the code embedded in the web page.

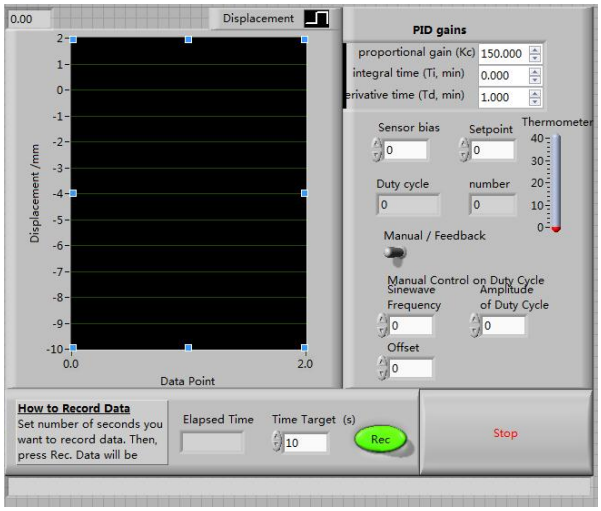

Figure 3: Original LabVIEW Front Panel of Nano Beam Experiment

When the user loads the web page in the beginning, the user interface is not currently connected with the web server. Figure 3 shows the original LabVIEW interface front panel, while Figure 4 shows the new interface. Upon pressing the “Start” button located on the JavaScript front panel, the embedded code initializes a connection with the web server that verifies if the user is

authorized to use this connection. Upon confirmation, the web server sends back to the client the acknowledgement and sends the “Start” command to the experiment. The JavaScript interface includes a timer that requests the live data formatted in JSON from the web server every half second. The user may choose to disconnect from the experiment at any time using the “Stop” button that stops the JavaScript timer and the experiment data acquisition.

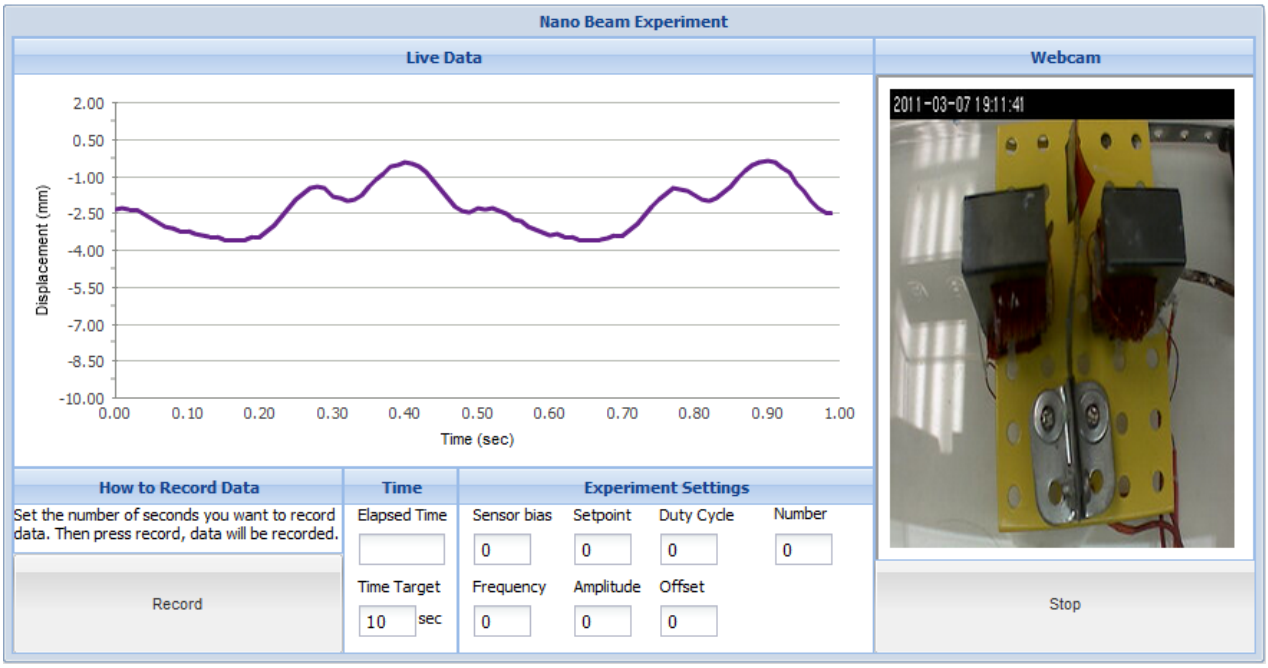

Figure 4: Interactive Media-rich JavaScript Experiment Interface

Table 2: Implemented REST methods for the retrieval of experiment and user data.

Client Request			**Server Response**	
Resource	**Parameters**	**Method**	**Action**	**Response Data**
Experiment				
/exp	/expID	GET	Gets all experiment information	Returns all experiment information JSON
User				
/user	/userID	GET	Gets all user information	Returns all user information JSON
Client Request			**Server Response**	
Resource	**Parameters**	**Method**	**Action**	**Response Data**
Experiment Data				
/expLive	/expID	GET	Get Data Array from Exp.	Returns Data Array JSON
/expLive	/expID /userID /active	POST	Case: {1-Start, 0-Stop} Update database. Inform Experiment.	Returns acknowledgement

The web server code was written using Python language scripting for rapid prototyping while maintaining a wide compatibility with server technologies. Table 2 shows the implemented REST methods for retrieving data from the online experiments. Each method allows the client program to request the corresponding data from the web server. The script serves the client interface. On the other hand, a PHP script allows the experiment server to save the live data in the respective web server database table.

Conclusion

The existing LabVIEW-based Nano Beam experiment at the University of Houston was converted to use the newly developed unified framework based on open and freely available software. The client JavaScript interface was developed to replicate the LabVIEW interface behavior. On the other hand, the LabVIEW experiment was modified by the simple addition of a start and stop sequence with the capability of forwarding the captured data over the network. Finally, a web server was setup to allow communication between the client and the experiment while maintaining a high level of security.

This unified framework for remote laboratory experiments was designed to allow performing online experiment from any computer and with no specific requirements. Anyone with an Internet connection and access to a web browser can interact with and control a remote experiment from anywhere. Users and experiment developers no longer have to worry about version problems or updates, since all the interfaces do not use proprietary technologies.

Acknowledgement and Disclaimer

This work is partially supported by the National Science Foundation under Grant Numbers EEC-0935208, EEC-0935008.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Bibliography

1. Ambrose, S. A., & Amon , C. H. (1997). Systematic design of a first-year mechanical engineering course at Carnegie Mellon University. Journal of Engineering Educations, 173-181.
2. Boehringer, D., Jeschke, S., & Richter, T. (2009). Lila - A European Project on Networked Experiments. Paper presented at the Sixth International Conference on Remote Engineering and Virtual Instrumentation.
3. Corter, J., Nickerson, J., Esche, S., & Chassapis, C. (2004). Remote vs. Hands-On Labs: A Comparative Study. Paper presented at the 34th ASEE/IEEE Frontiers in Education Conference.
4. Duro, N., Dormido, R., Vargas, H., Dormido-Canto, S., et al. (2008). An Integrated Virtual and Remote Control Lab: The Three-Tank System as a Case Study. Computing in Science & Engineering, 10(4), 50-59.
5. Felder, R. M., & Brent, R. (2005). Understanding Student Differences. Journal of Engineering Education, 21(1), 166-177.
6. Jeschke, S., Richter, T., & Sinha, U. (2008, Oct. 2008). Embedding Virtual and Remote Experiments Into a Cooperative Knowledge Space. Paper presented at the 38th ASEE/IEEE Frontiers in Education Conference, Saratoga Springs, NY.

7. Jing, M., & Jeffrey, V. N. (2006). Hands-on, simulated, and remote laboratories: A comparative literature review. ACM Comput. Surv., 38(3), 7.
8. Olmi, C., Song, G., & Mo, Y. L. (2007). An innovative and multi-functional smart vibration platform. Smart Mater. Struct., 16, 1302–1309.
9. Song, G., Olmi, C., & Bannerot, R. (2007). Enhancing Vibration and Controls Teaching with Remote Lab Experiments. Paper presented at the Proceedings of the 2007 ASEE Annual Conference & Exposition.