QSCIENCE PROCEEDINGS

OPEN ACCESS

World Congress on Engineering Education 2013

Development of a remote experiment under a unified remote laboratory framework

Xuemin Chen[1], Daniel Osakue[1], Ning Wang[2], Hamid Parsaei[3,*], Gangbing Song[4]

[1]Department of Engineering Technology, Texas Southern University, Houston, TX 77004, USA
[2]Department of Computer Science, Texas Southern University, Houston, TX 77004, USA
[3]Department of Mechanical Engineering, Texas A&M University at Qatar, Doha, Qatar
[4]Department of Mechanical Engineering, University of Houston, Houston, TX 77204, USA
*Email: hamid.parsaei@qatar.tamu.edu

ABSTRACT

In this paper, a Smart Vibration Platform (SVP) remote experiment under a novel unified framework for implementing remote laboratory is presented. In this unified remote laboratory framework, a Comet solution via `Node.js` and its `Socket.IO` package was implemented on the server side. A new web socket protocol, which lets the experiment to communicate with Socket.IO, was created for the workstation. With this unified framework, the user can remotely conduct the experiment by using any portable device without installing any plug-in.

Keywords: remote experiment, emerging educational techniques, engineering education

http://dx.doi.org/10.5339/qproc.2014.wcee2013.7

Submitted: 18 January 2014
Accepted: 30 April 2014

QSCIENCE.com
An Initiative of Qatar Foundation

Cite this article as: Chen X, Osakue D, Wang N, Parsaei H, Song G. Development of a remote experiment under a unified remote laboratory framework, QScience Proceedings (World Congress on Engineering Education 2013) **2014:7** http://dx.doi.org/10.5339/qproc.2014.wcee2013.7

1. INTRODUCTION

The challenge of developing cross-browser and cross-device web user interfaces has shown the need for a unified remote laboratory framework.[1–3] Based on Web 2.0 technology,[4] we developed a unified remote laboratory framework built directly on top of; a database, HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and jQuery/jQuery-Mobile JavaScript libraries. The PHP (which stands for PHP Hypertext Preprocessor) and database-driven data-streaming solution eliminated the need for the client side LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) plug-in and thus vastly improved accessibility through a pure Web 2.0 interface. This ensures minimal functionality across most common web-browsers and browser-capable devices, such as tablets and smart phones. However, maintaining a consistent and usable interface across different web-browsers and devices has proven to be time-consuming and challenging. In addition, we found that there are some unreliable and unstable problems in the LabVIEW database connection functions, which were used in earlier proposed solutions.[1–3] Since we cannot access the source codes in the libraries for LabVIEW database connection, we cannot identify the root cause. To solve these problems, we propose using a Comet[5] solution via `Node.js` and its `Socket.IO` package on the server side. We also create a partial implementation of the Web Socket protocol for our workstation programs and experiments to communicate with `Socket.IO`.

2. METHODOLOGY

The goal of this work is to provide a remote experiment interface that will work with any Internet-enabled web browser, on any device, without the need to install any software or plug in. We propose a new unified framework for conducting remote experiments over Internet. In this work, we mainly use three technologies for the system implementation; `LabVIEW` to `Node.js` technology, for experiment data transmission and experiment equipment control commands; video transmission technology, for real-time system monitoring; and mashup technology for user interface implementation.

The system architecture is shown in Figure 1. We divide the system architecture as three parts: client, server, and experiment equipment control.

We use three different servers to handle the data transmission through three different network ports. The user interface framework is generated by Apache web server via port 80. The experiment data and experiment equipment control commands are transferred by the `Node.js` server via port 1029. And the video is transferred by video server via port 1026. We use the mashup technology[6] to combine the experiment result display panel, experiment equipment control panel, and experiment real time video display panel in the end user interface.

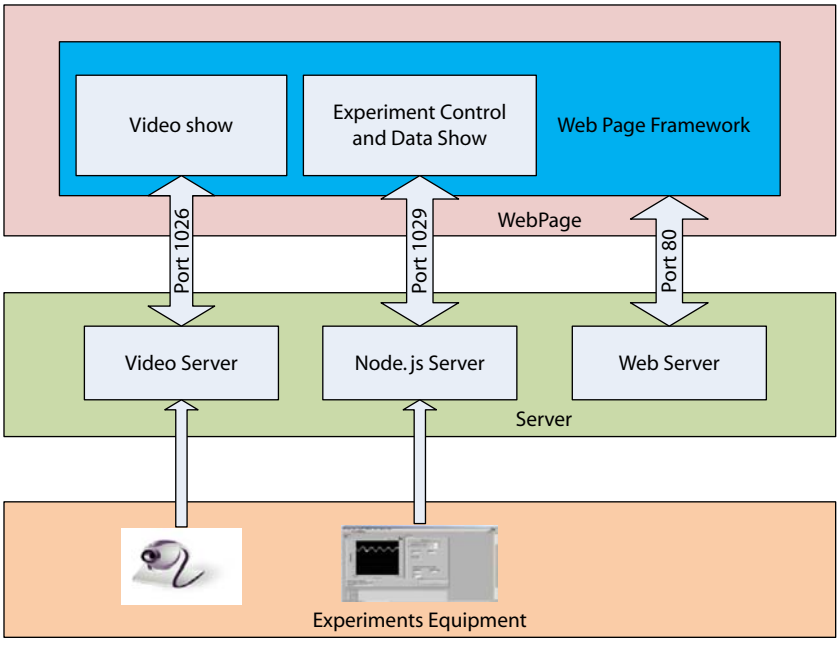

Figure 1. System architecture.

2.1 The LabVIEW to Node.js technology for the server implementation

On the server side, we propose a LabVIEW to Node.js (LtoN) technology to achieve the experiment data transmission and experiment control commands transmission. The LtoN is one of the three technologies we used for developing plug-in free, web-based remote experiment with LabVIEW. LabVIEW is a system design platform and development environment for a visual programming language from National Instruments (NI). LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, various versions of UNIX, Linux, and Mac OS X.

Node.js is a software package designed for writing highly-scalable internet applications, notably web servers.[7] The programs of Node.js are written in JavaScript, and Node.js uses an event-driven, non-blocking I/O model.[7] This means in Node.js there's no way to block the currently working thread. Every operation in Node.js is executed asynchronously. It has huge benefit especially for the code needs I/O operations such as reading disks, connecting to database, and consuming web service, etc.

A reference working flow of that only one working thread serves all users' requests and resources response in Node.js architecture is shown in Figure 2.

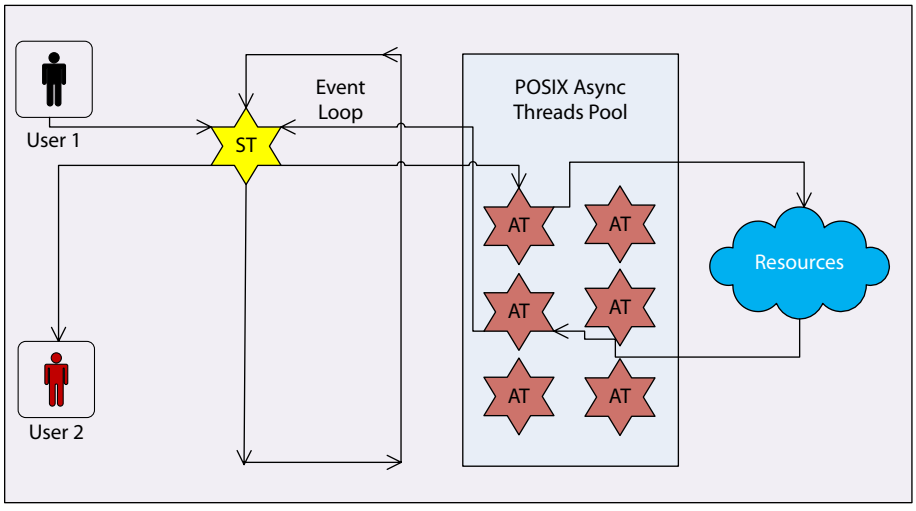

Figure 2. Node.js **architecture.**

The LtoN solution is suitable for developing the virtual and remote laboratories which employ the National Instruments LabVIEW programming language and data acquisition systems. It can relay sensor or instrumentation data from LabVIEW to the client web-page and passes control data instructions from the client web-page to LabVIEW.

The design principles of LtoN are:

- Minimal configuration and setup
- Maximum reliability and robustness
- Web-standards compliance
- Fast data streaming with the JSON format
- Client-based processors need only a browser
- Work with any LabVIEW data type that can be converted to a string

In the remote experiment framework proposed in Osakue et al.,[3] the experiment data needs to follow these steps; updating data in the database, then retrieving and converting the data to JSON format, and finally sending to the client with PHP. After that, all of the experiment data are sent to the end user in a browser.

The new framework for developing remote laboratory system results to increase system efficiency, simplify LabVIEW code, and improve system stability. In the new unified framework, the data are directly converted to JSON by LabVIEW and then sent to Socket.IO, which is handled in the client. Finally we combined all of data in the web browser via mashup technology, and then presented the

data to the end users directly. We also saved the data collected by LabVIEW to the database for backup.

The levels of the new framework and the technology/protocol/software are showed in Table 1.

Table 1. Technology/ protocol/software list for the new framework.

Level	Name	Technology/Protocol/Software	Remark
1	Client-user Interface	Mashup technology, JavaScript,	
2	Data Protocol	Socket.IO/web socket	
3	Server-Web Service	Node.js (Vo.8.8), JSON, MySQL,	LtoN
4	Experiment Server	LabVIEW (V8.6)	

The working process of the new framework based on the LtoN technology is shown in Figure 3.

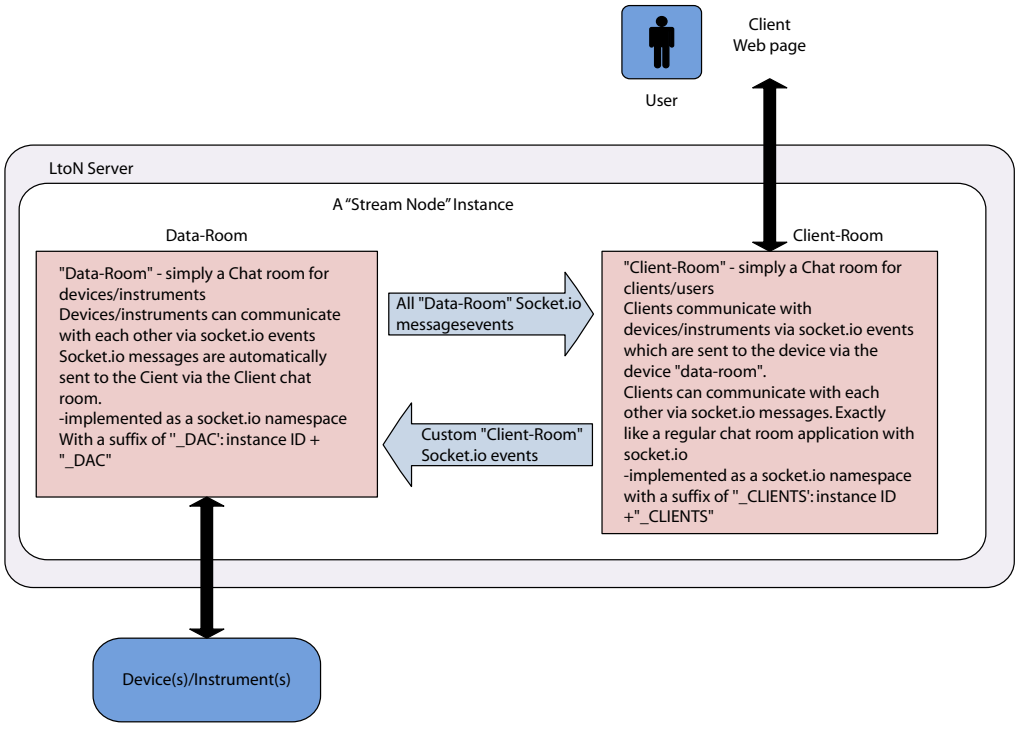

Figure 3. LtoN server.

2.2 Real-time video transmission technology

In the real-time video transmission part, we directly use a network camera to collect real-time video, and then transfer the video stream from video server directly via port 1026 to the JW Player (Version 5.0) in the browser. We used the JW Player in a web page which supports "auto-start" in PC for different browsers like IE, Chrome, Safari, Firefox, etc. to show the real-time video for the end user.

For the video streaming, Video LAN Client (VLC) media player/server for streaming to Apple devices and C++RTMP server is used by LtoN to stream video to non-Apple web browsers/devices that support Flash video. C++RTMP server is an open source RTMP streaming server used by LtoN to service streaming video connections that use the RTMP protocol. VLC is an open source multimedia solution is use to service streaming video connections that require the HTTP Live Streaming (HLS) method. HLS is the Apple Inc.-mandated video streaming method for recent, current and future Apple devices. Only VLC versions 2.0.1 or higher are capable of HLS. Currently our implementation requires a compiled version of the C++RTMP server, but we are planning to switch to the repository-installed version to simplify overall installation of the LtoN. The video transmission process is shown in Figure 4.

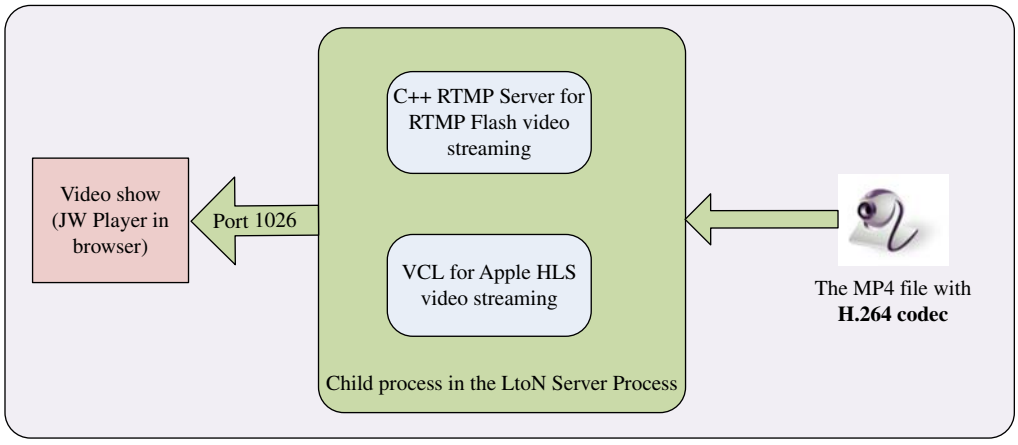

Figure 4. The real-time video transmission.

On the client side, we used the JW Player for Flash, which is a flexible and popular media player, since it supports playback of any format the Adobe Flash Player can handle, as well as HTTP and RTMP streaming and various XML play list formats. A wide range of configuration options can be set, and an extensive JavaScript API is available.

The JW Player supports many different video formats like MP4 (.mp4, .m4v, .f4v, .mov), WebM (.webm), FLV (.flv), OGG (.ogv). In our framework we selected the MP4 format, because it is today's format of choice, supporting the best video quality and hardware-accelerated playback on a wide range of browsers and devices. The files in the MP4 container format, containing video encoded with the H.264 codec and audio encoded with the AAC codec, and the network camera which we used, can directly support H.264 video format collection.

A list of browsers and their supporting modes are shown in Table 2. Based on the Table 2, we know that the JW Player can be run on the different browsers in all popular platforms.

Table 2. List of browsers and their supporting modes.

Browsers	Support mode
Internet Explorer 6/7/8	Flash mode
Firefox	Flash mode (not in HTML5)
Chrome	Flash, HTML5 modes
Internet Explorer 9	Flash, HTML5 modes
Safari	Flash, HTML5 modes
Opera	Flash mode (not in HTML5)
iOS (iPad/iPhone)	HTML5, Download modes
Android	Flash, HTML5, Download modes
BlackBerry	Download mode

So far the biggest problem we have faced using JW Player, is that it can't support "auto-start" on smartphone platforms like iOS, Android, and Windows Mobile etc. In addition, it needs to install the plug-in in the smartphone platform to run this player, since JW Player is based on Flash technology.

2.3 The mashup technology for the user interface implementation

To implement the end user interface, we selected the mashup technology to combine the contents which came from the different servers via the different ports. A mashup is a web page, or a web application, that uses and combines data, presentation or functionality from two or more sources to create new services.

The architecture of a mashup is divided into three layers:

- Presentation/user interaction: this is the user interface of mashup. The technologies used are HTML/XHTML, CSS, JavaScript, Asynchronous JavaScript and XML (Ajax).
- Web services: the product functionality can be accessed using API services. The technologies used are XML-RPC, JSON-RPC, SOAP, REST.
- Data: handling, sending, storing and receiving data. The technologies used are JSON, XML.

In general, there are two styles of mashup: Web-based and server-based. Whereas web-based mashups typically use the user's web browser to combine and reformat the data; server-based mashups analyze and reformat the data on a remote server and transmit the data to the user's browser in its final form.

The mashup architecture is shown in Figure 5.

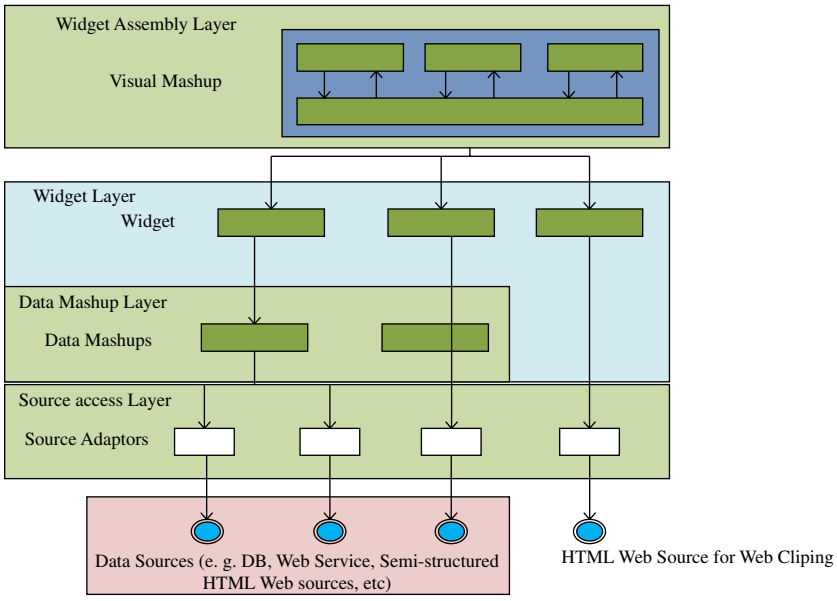

Figure 5. Mashup technology.

In our current framework, we mainly use web-based mashup technology in the browser for data combination for the end users. The end user interface developed for the Smart Vibration Platform (SVP) remote experiment under the proposed unified frameworks is shown in Figure 6.

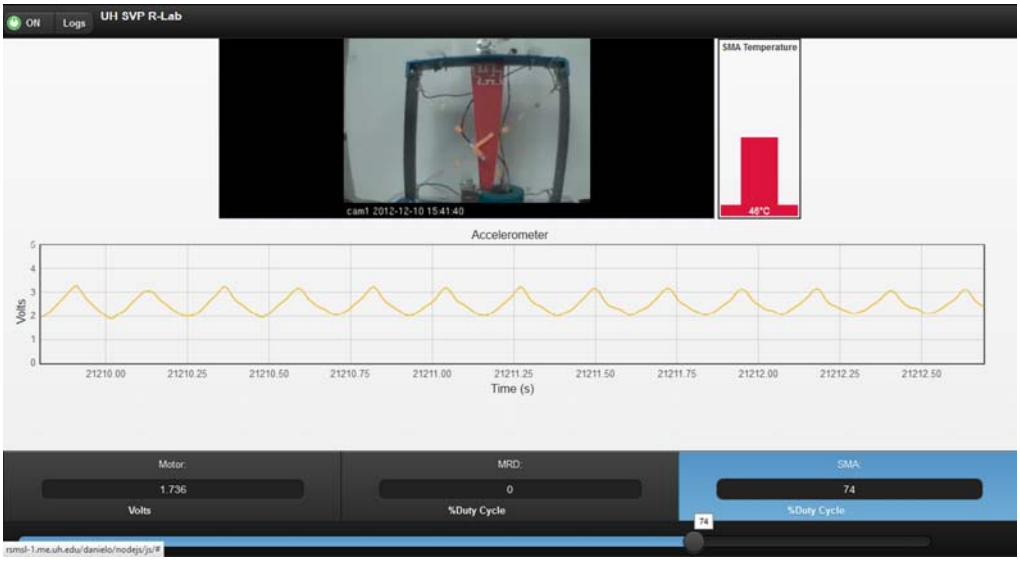

Figure 6. End user interface developed under the proposed unified frameworks.

This user interface can be run any PC or Mac computer without installing any plug-in or software. When the switch on the left corner is on, the user can the slide bar to control the motor rotation speed. The accelerometer output, which indicates the platform vibration, is shown in the middle of the screen. Real-time video is displayed at the top. The platform vibration can be controlled by the magneto-rheological damper (MRD) and the smart material alloy (SMA). When the user turns on the SMA, the temperature of SMA wire will rise. This is shown on the right side of the video.

3. FIREWALL ISSUES AND FUTURE WORK

In our new framework, we increase system efficiency, simplify LabVIEW code, and improve system stability. However, there is another problem which needs to be solved: the firewall issue. Since we use three ports (ports 80, 1026 and 1029) to transfer the different data to the user interface, it may cause firewall issues in some circumstances. We used the newly-developed remote SVP experiment at TAMU Qatar. Only port 80 of TAMU Qatar firewall is open. Without using the University of Houston virtual private network (VPN), the users can't see the real-time video or accelerometer output, and can't control the SVP.

To solve this problem, we plan to use only one network port, port 80, to transfer all data. So we will a server-based mashup in this instance. We will combine all data on the server side, and then send the combined data package to the user interface via port 80.

4. CONCLUSION

We have presented a new unified framework for remote experiment development. A remote SVP experiment was successfully developed under this new unified framework. The user interface can be run by any browser, on any platform, without installing any plug-in. However, it may have a firewall issue if the ports 1026 and 1029 are blocked. Real-time video can't automatically start on the portable devices without Flash installed. The solution of these issues is to use server-based mashup technology.

Acknowledgements

This material is based upon work supported by the Qatar National Research Fund under Grant No. NPRP 4-892-2-335.

REFERENCES

[1] Osakue D, Chen X, Ahmed O, Darayan S, Olowokere D. Virtual and remote laboratory framework development for engineering technology education - a case study. In *Proceedings of ASCE Earth and Space 2012*, pp.1211–1217, Pasadena, California, April 15–18, 2012.
[2] Cao B, Song G, Chen X, Osakue D. Platform independent interface for remote laboratory experiments. In *Proceedings of ASEE Annual Conference & Exposition*, San Antonio, Texas, June 10–13, 2012.
[3] Osakue D, Chen X, Wang C, Ahmed O. Develop a cross browser compatible DSP remote laboratory with zero plug-in installation. In *Proceedings of ASEE Annual Conference & Exposition*, San Antonio, Texas, June 10–13, 2012.
[4] Wikipedia, "Web 2.0 – Wikipedia, the free encyclopedia," Online, http://en.wikipedia.org/wiki/Web_2.0, accessed November 1, 2012.
[5] Wikipedia, "Comet – Wikipedia, the free encyclopedia," Online, http://en.wikipedia.org/wiki/Reverse_Ajax accessed November 1, 2012.
[6] Wikipedia, "Mashup – Wikipedia, the free encyclopedia," Online, http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid), accessed November 1, 2012.
[7] Wikipedia, "Node.js – Wikipedia, the free encyclopedia," Online, http://en.wikipedia.org/wiki/Node.js, accessed November 1, 2012.